Digital design lab 4A

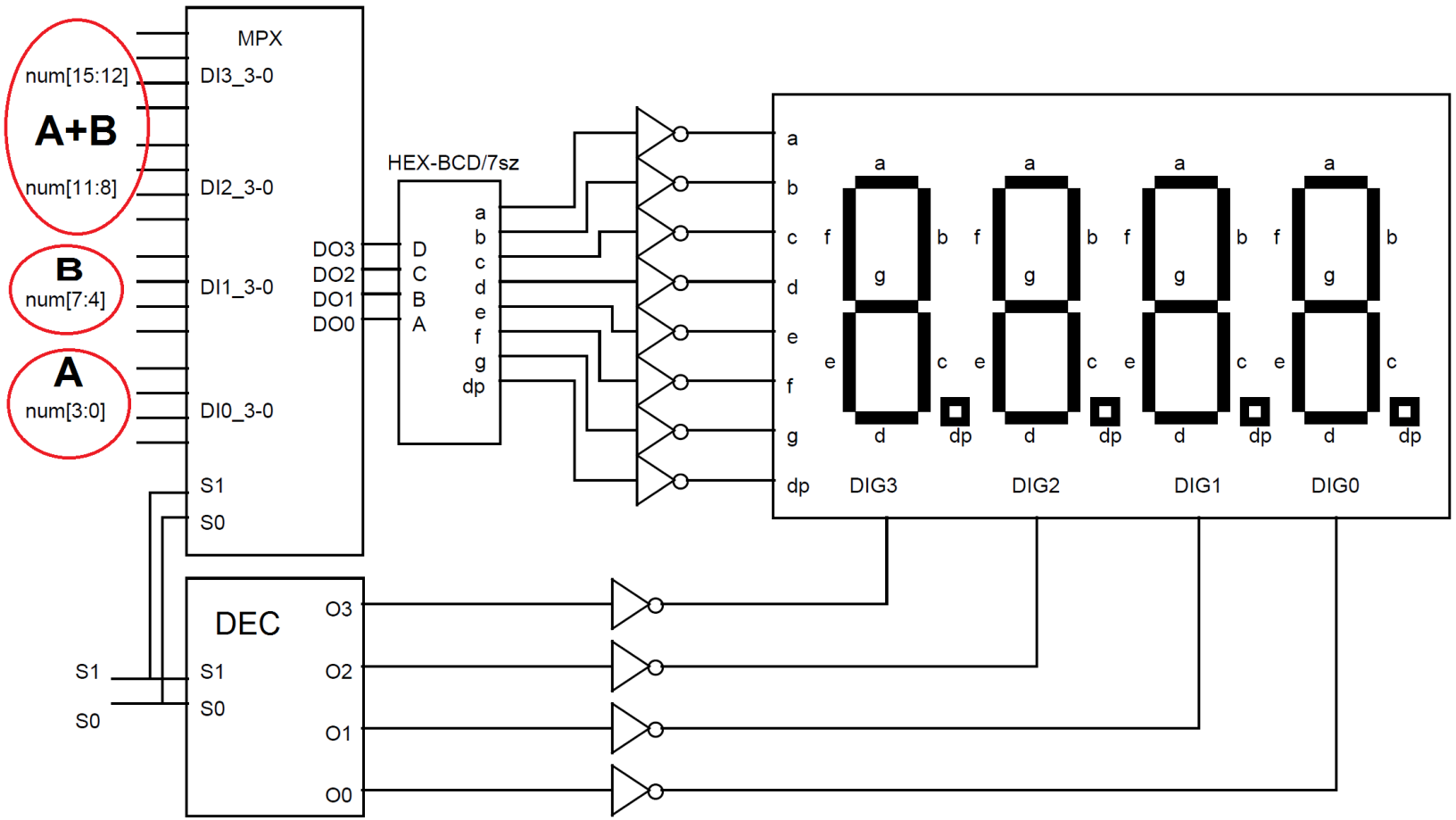
Digital design lab 4A

- Today you will learn about hierarchical design of digital circuits
- The design will be done on 2 levels:
  - First you will design the submodules
  - Then the top module will be designed, and the submodules will be instantiated
- The PC is good example for hierarchical digital systems: the submodules are the processor, memory, VGA, etc. Each submodule has a well defined functionality, they are independent subsystems with unique behavior.
- In this case the top module is the motherboard, which connects the submodules, and integrates the subsystems (submodules) in to one high-level system.

Digital design lab 4A

- The specification of the high level system of today's laboratory is the following: the circuit reads two hexadecimal numbers from the switches: one number from sw[3:0], and the other from sw[7:4].
- The two numbers (A and B), and their sum (A+B) will be shown on the 7-segment display of the system: one digit for A, one digit for B, and two digits for A+B (since the maximum value is **F + F = 1E**).
- To implement such system, we need the following components: multiplexer, decoder, hexadecimal to 7-segment converter.
- In addition, we need a circuit that continuously counts from 0 to 3. This is a sequential circuit, so it won't be studied in details right now.

- The design is the following:



Digital design lab 4A

- Let’s start with the 7-segment display (in the top right corner). It has two group of inputs:
  - Inputs a b c … g and dp: these inputs control the LEDs, dp is for the decimal point LED, it will not be used today.
  - Inputs DIGn: Digit selector inputs, the LED control inputs are forwarded to the selected digit.
- These are **negative logic inputs:** 0 input means true and 1 means false.
- **Example**: if the b and c inputs are 0, the others are 1 on the LED control input, and DIG2=0, the others are 1 on the digit selector input, than a 1 will appear on the second digit from the left.
- **Note: we will design the circuit assuming positive logic, and we will invert the signals in the last step (**as it is shown in the figure**)**

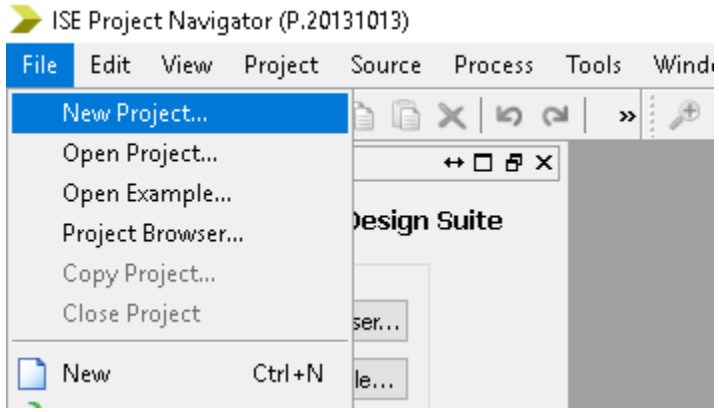
Digital design lab 4A

- The S1-S0 signals belong to the counter that counts from 0 to 3 repeatedly: 00, 01, 10, 11, 00, 01, 10, 11, 00, …..
- We use the decoder to decode this signal: the corresponding output of the decoder is set to 1 to select a digit.
- At the same time S1-S0 is the input of the multiplexor too: based on the value of S1S0, the multiplexor forward one of its 4 4-bit input.
- This value is converted to a 7-segment number, and the result of the conversion appears on the LED control inputs of the 7-segment display.
- Let’s start the implementation.

Digital design lab 4A

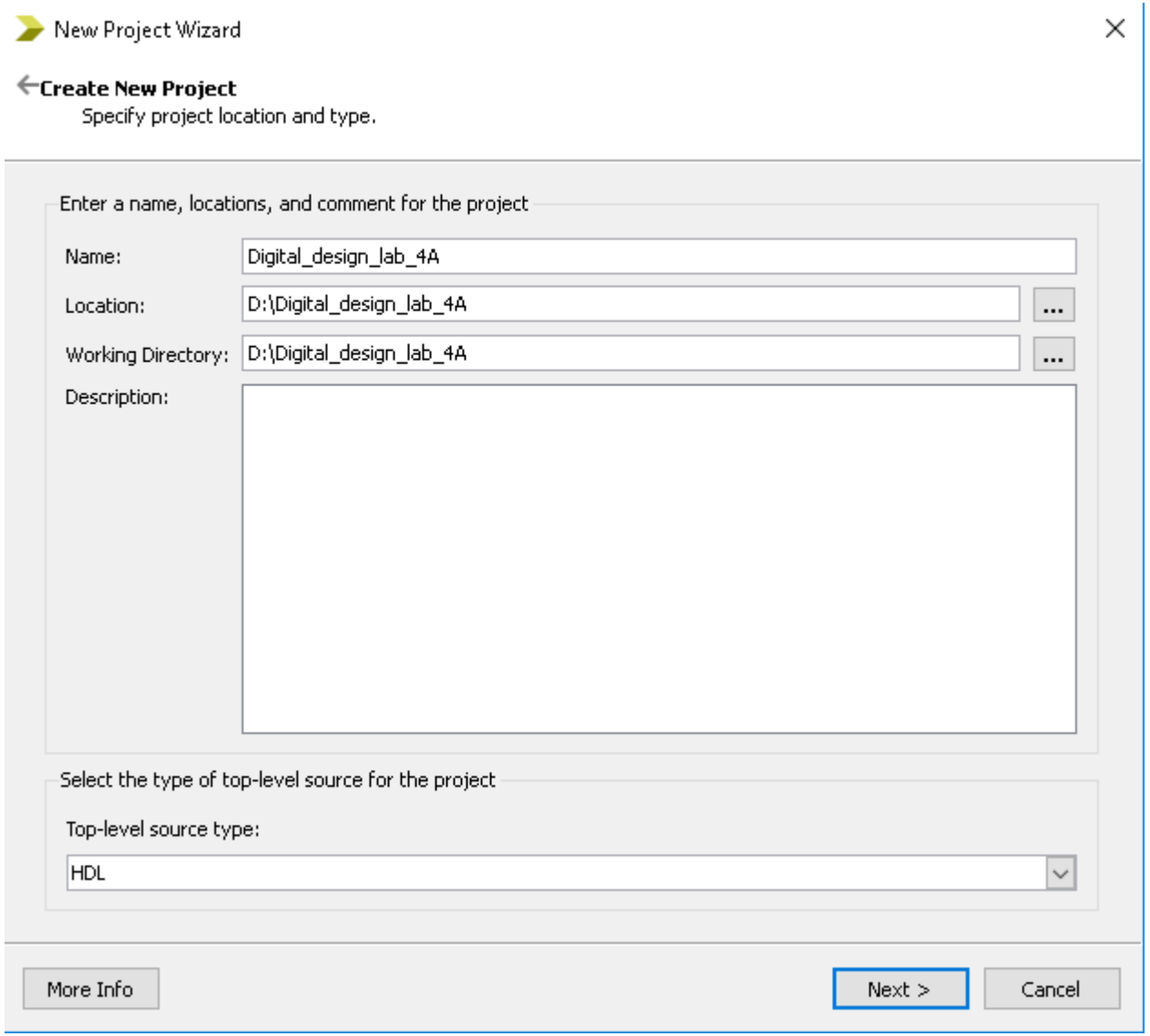
- Launch the ISE design suite

- Create a new project



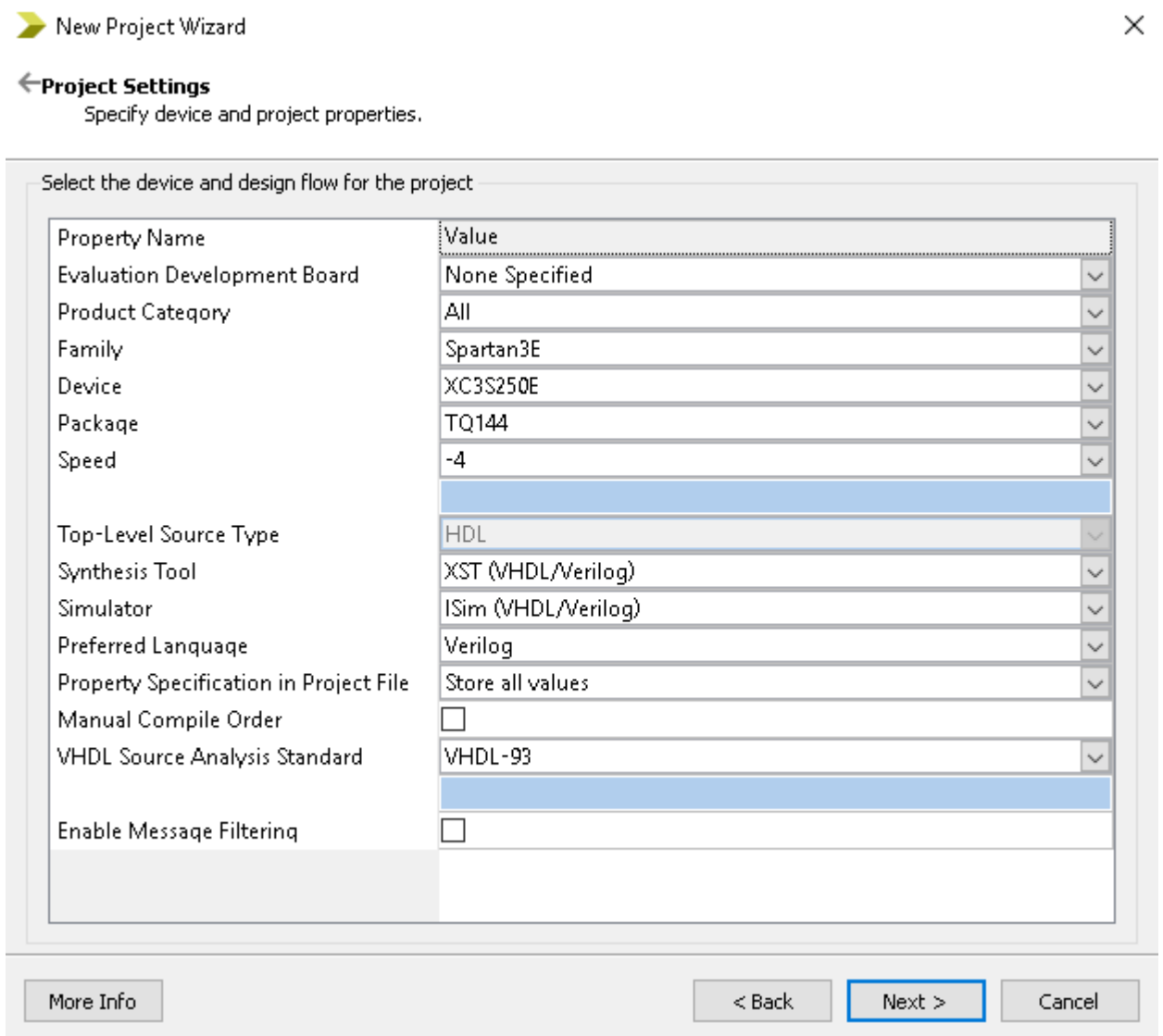
Digital design lab 4A

- Name: Digital_design_lab_4A
- Work on the D: drive
- Press Next



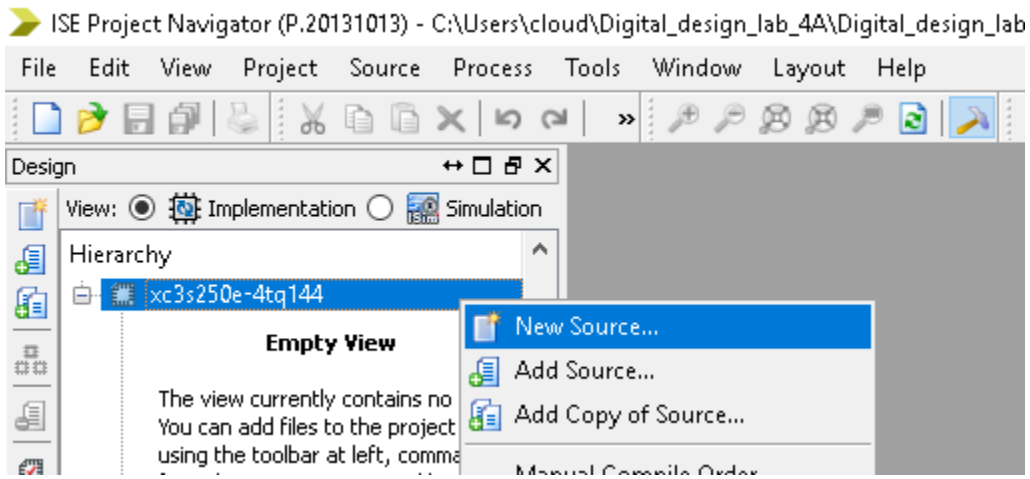
Digital design lab 4A

- Verify the settings:
  - Spartan3E
  - XC3S250E
  - TQ144
  - -4
- Press Next, than Finish



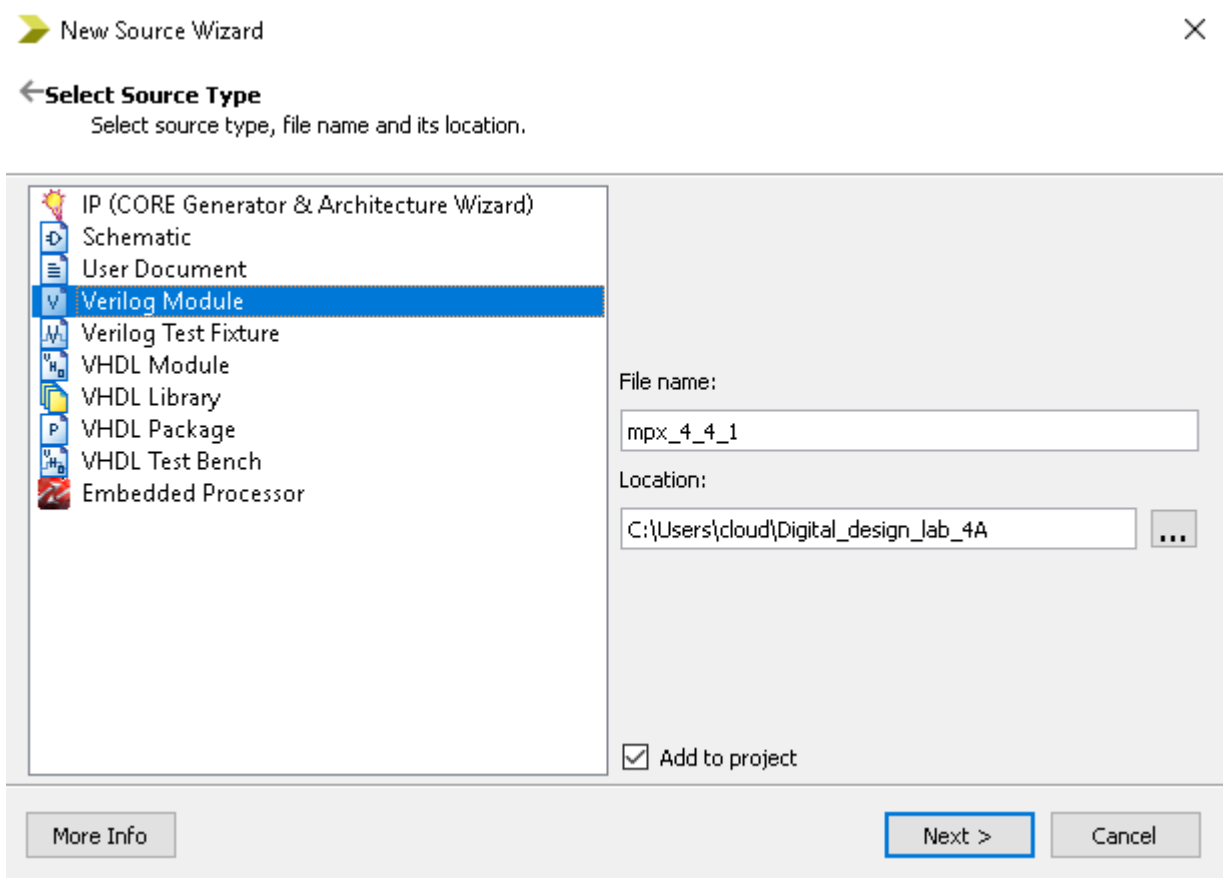
Digital design lab 4A

- Right click on the label
- Select New source…



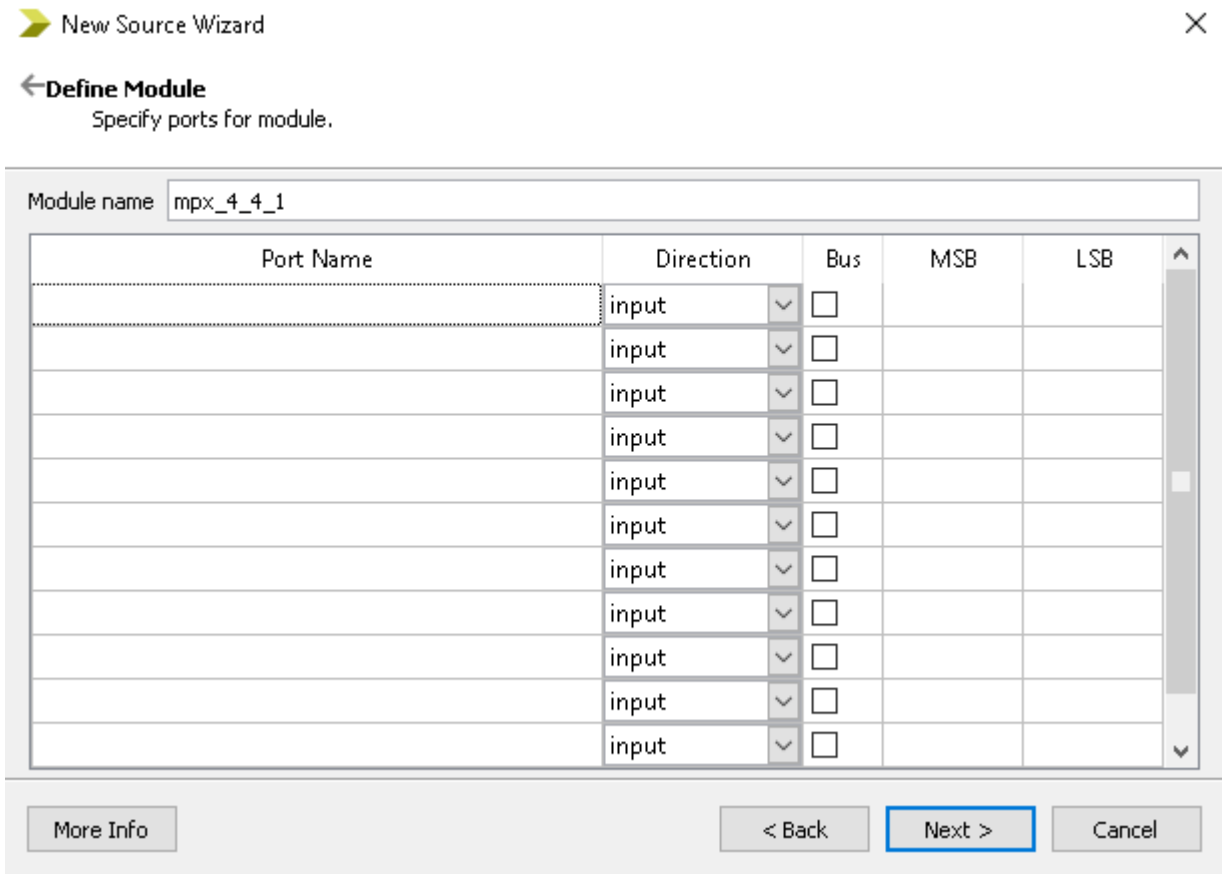
Digital design lab 4A

- Select Verilog module
- Name: mpx_4_4_1
- This will be a 4x4x1 multiplexor: It has 4 4-bit input
- Make sure “Add to project” is checked
- Press Next



Digital design lab 4A

- Do **NOT** add the signals here,
  We will add them manually later
- Press Next, then Finish



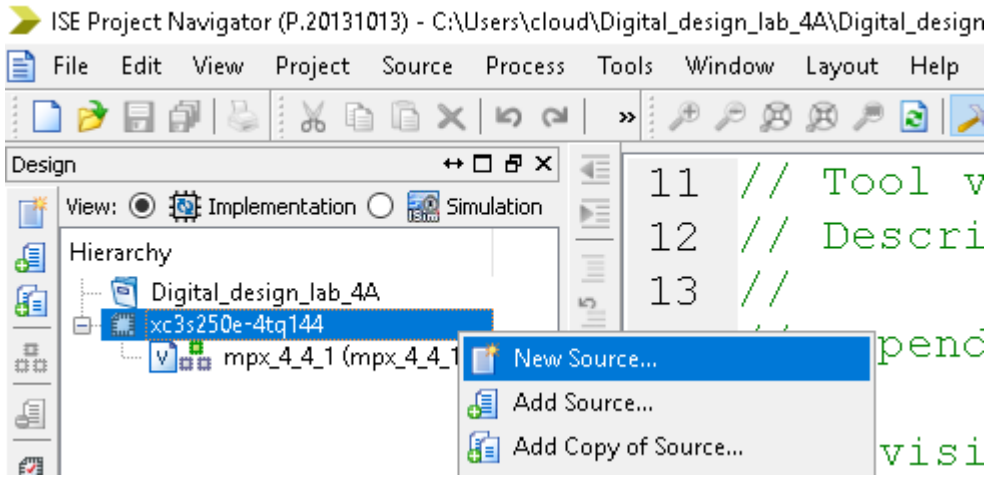
Digital design lab 4A

- Add the following source code:
- Note: we use “output reg” instead of “output” in the module declaration
- This is needed to be able to use the always block
- As we have seen on the practice, this is a simpler and more overviewable way to describe the circuit’s behavior.
- Use the “<=“ operator for register assignments (instead of “=“)
- Save all changes 

```
21 module mpx_4_4_1(
22     input [1:0] s,
23     input [15:0] num,
24     output reg [3:0] out
25     );
26
27 always @ (*)
28     case (s)
29         2'b00: out <= num[3:0];
30         2'b01: out <= num[7:4];
31         2'b10: out <= num[11:8];
32         2'b11: out <= num[15:12];
33         default: out <= num[3:0];
34     endcase
35
36 endmodule
37
```

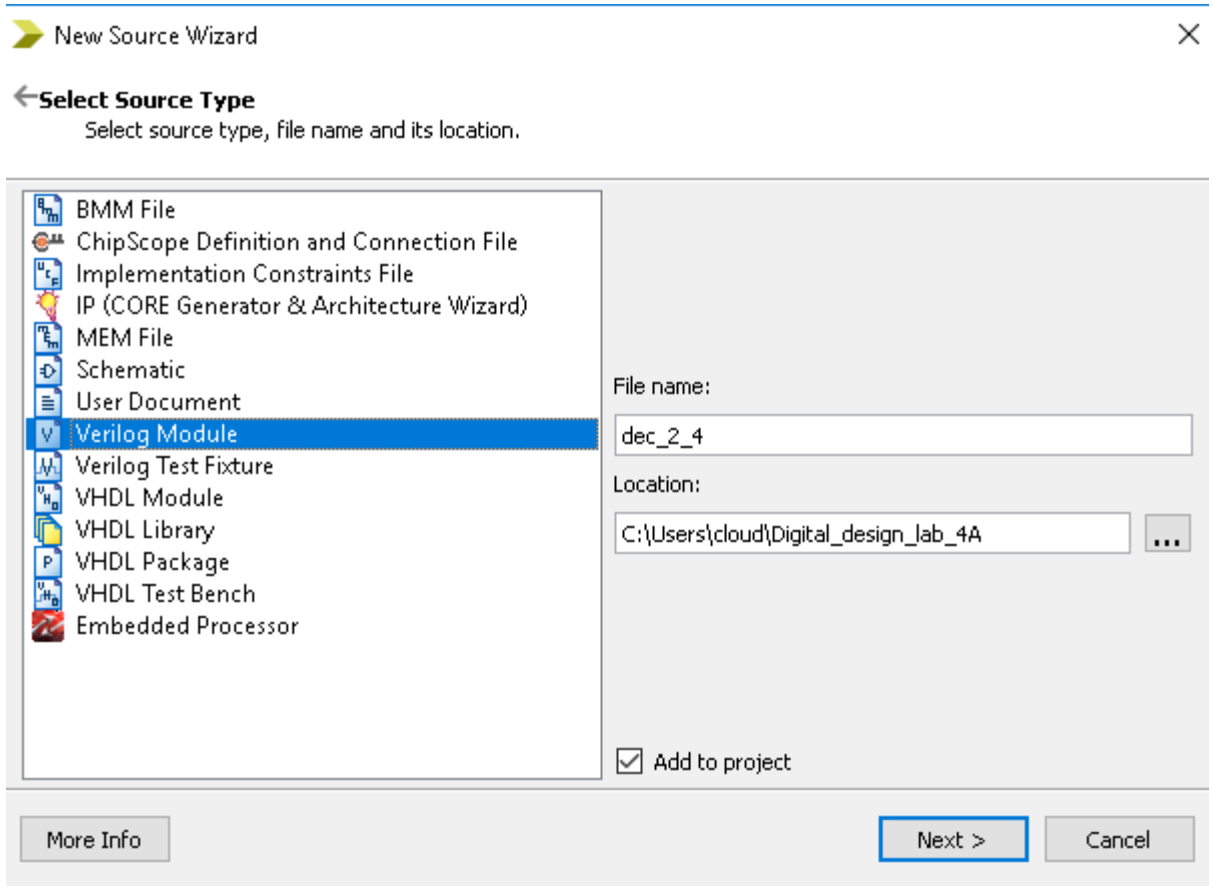
Digital design lab 4A

- Add another new source file



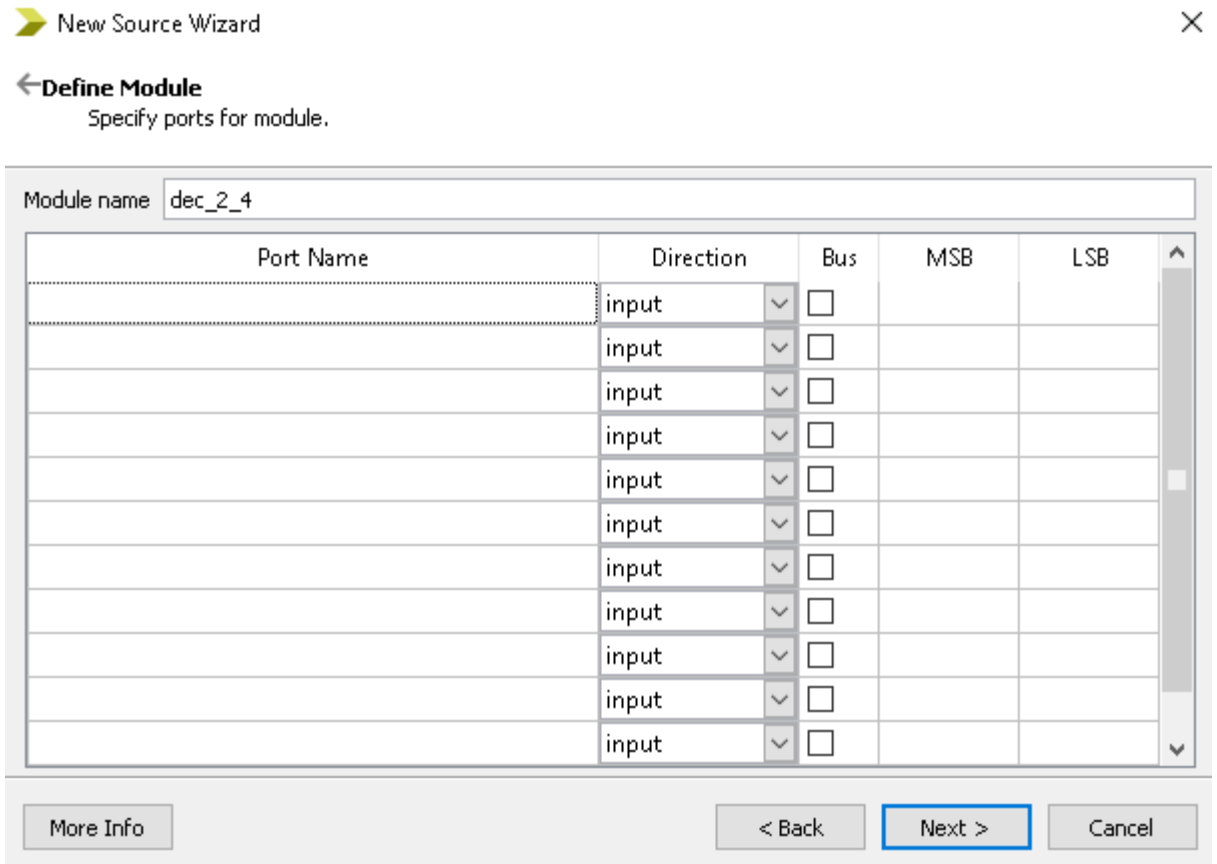
Digital design lab 4A

- Verilog module
- Name: dec_2_4
- This will be the decoder
- Make sure “Add to project” is checked
- Press Next



Digital design lab 4A

- Leave the table blank
- Press Next, then Finish



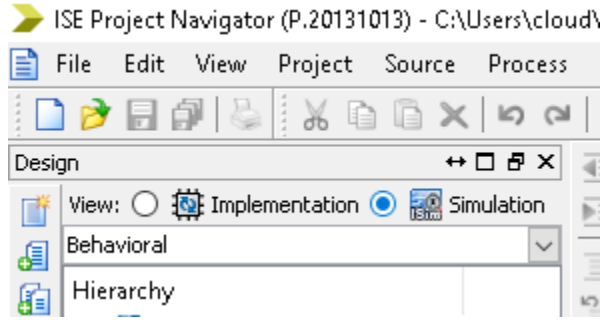
Digital design lab 4A

- Add the following source code:
- Mind the reg keyword after output in the header
- Save all changes 

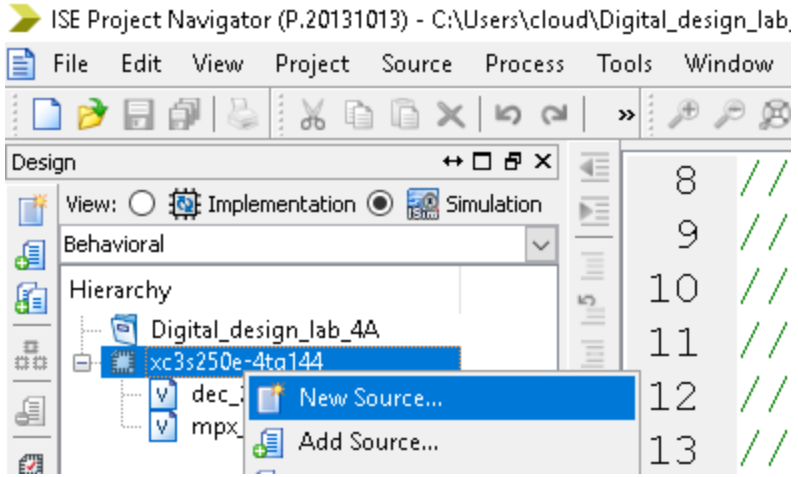
```
21 module dec_2_4(
22     input [1:0] s,
23     output reg [3:0] dig
24     );
25
26 always @ (*)
27     case (s)
28         2'b00: dig <= 4'b0001;
29         2'b01: dig <= 4'b0010;
30         2'b10: dig <= 4'b0100;
31         2'b11: dig <= 4'b1000;
32         default: dig <= 4'b0001;
33     endcase
34 endmodule
```

Digital design lab 4A

- Now we will test the decoder and the multiplexor
- First switch to Simulation mode

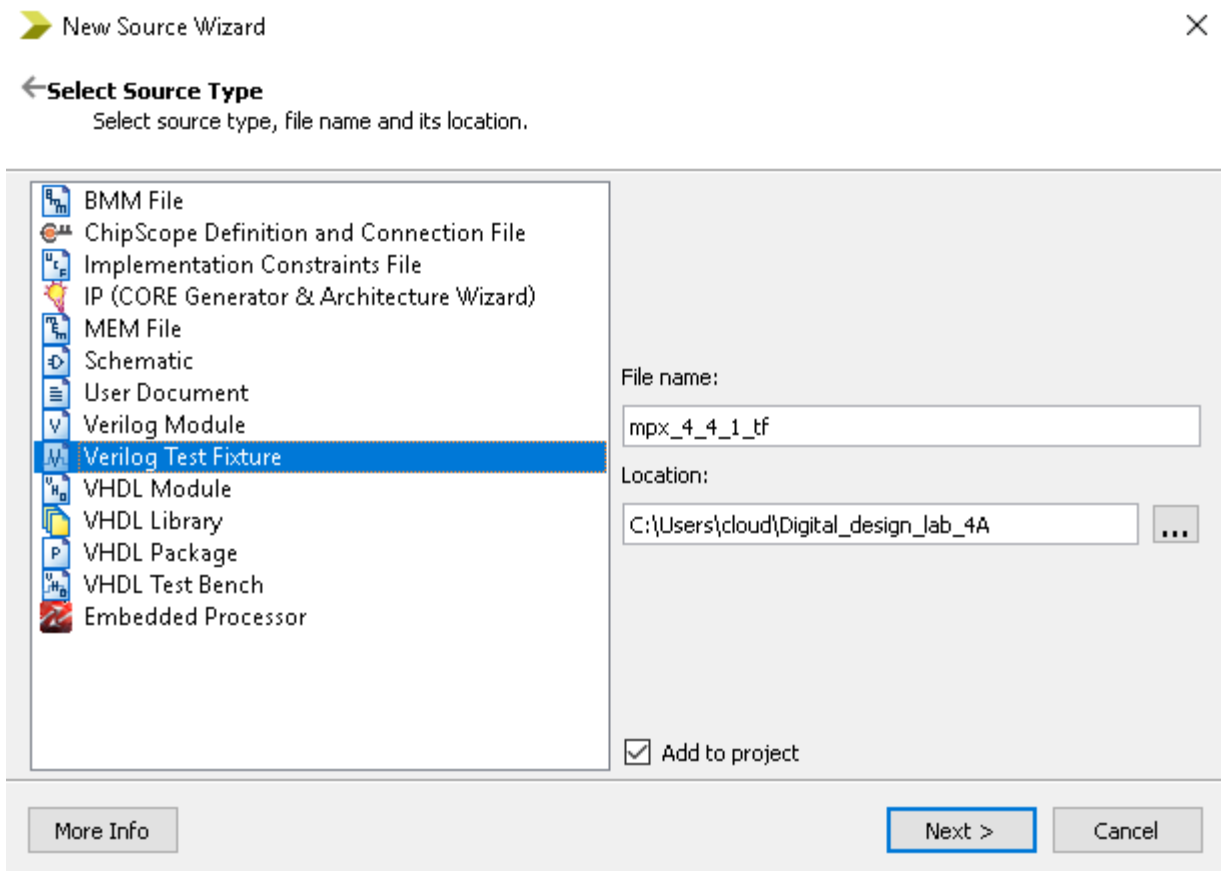


- Then right click on the label, and select New Source



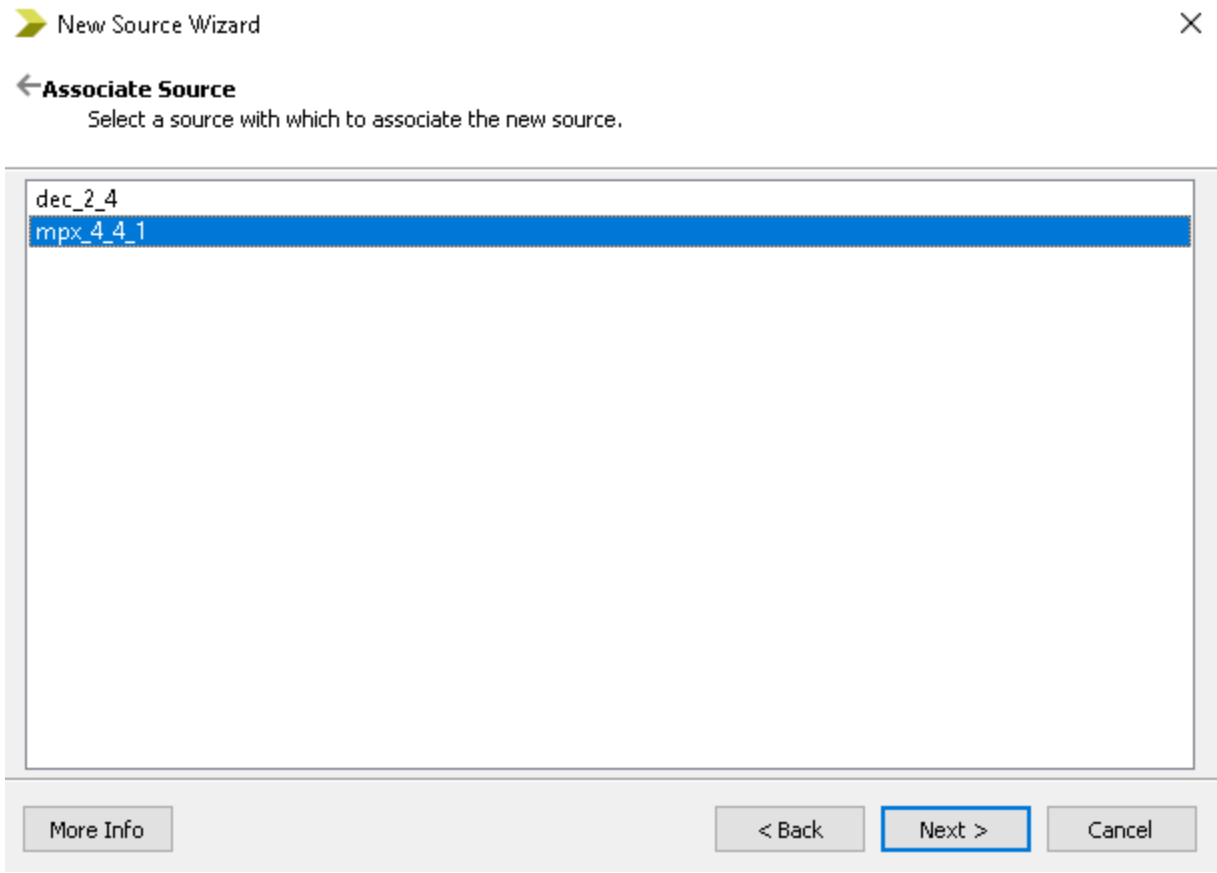
Digital design lab 4A

- Select Verilog Test Fixture
- Name: mpx_4_4_1_tf
- Press Next



Digital design lab 4A

- Select the mpx_4_4_1 circuit here
- Press Next, then Finish



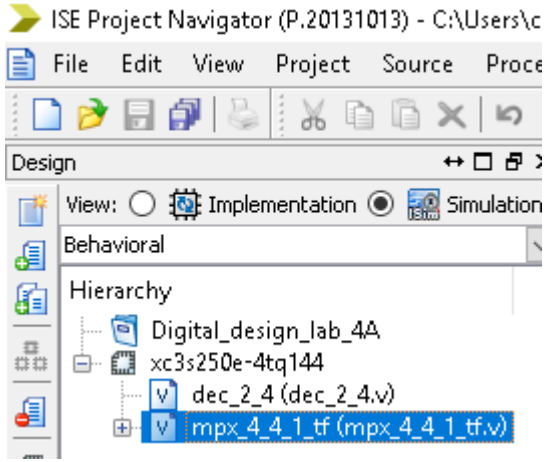
Digital design lab 4A

- Add the following code after the initial begin part
- Save all changes 

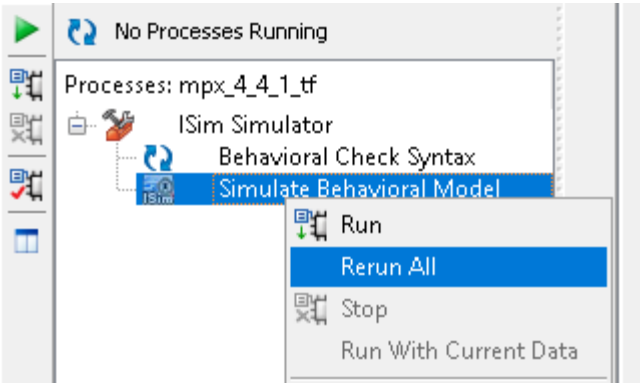
```
41      initial begin
42          // Initialize Inputs
43          s = 0;
44          num = 0;
45
46          // Wait 100 ns for global reset to finish
47          #100;
48
49          // Add stimulus here
50          #100; s=2'b00; num=16'h048C;
51          #100; s=2'b01;
52          #100; s=2'b10;
53          #100; s=2'b11;
54
55      end
```

Digital design lab 4A

- Left click on the mpx_4_4_1_tf module in the top left corner of the ISE

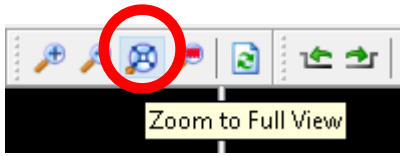


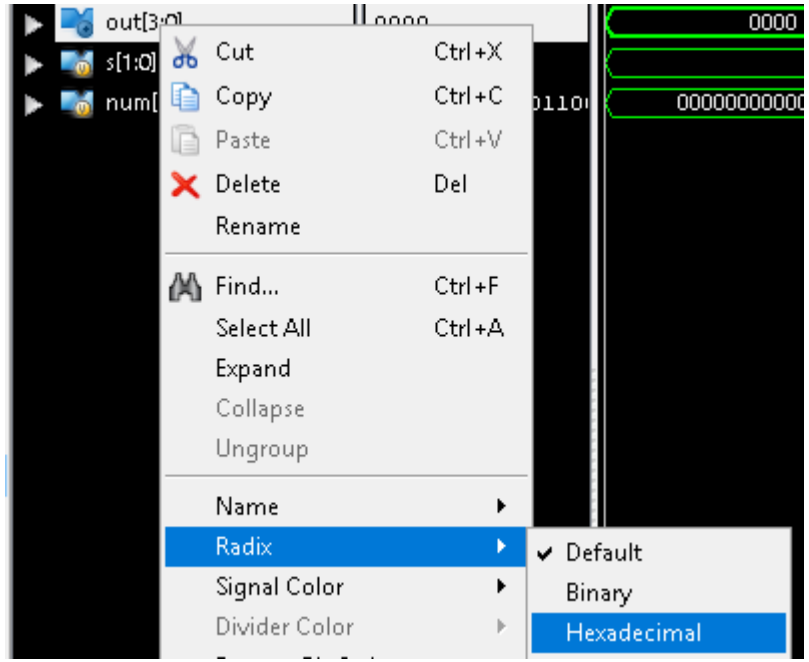
- Then right click on “Simulate Behavioral Model”, and select “Rerun All”



Digital design lab 4A

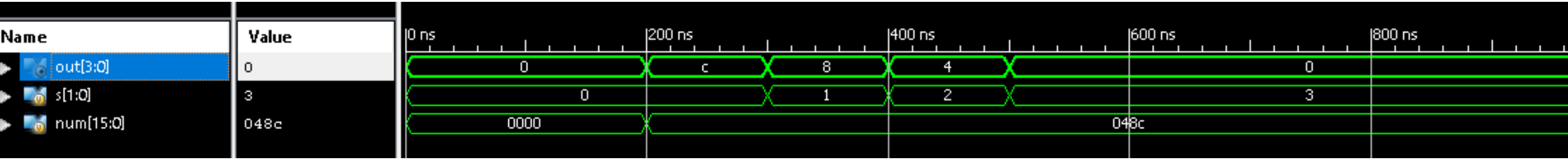
- A window pops up with the simulation results. Press the “Zoom to Full View” button
- Right click on the out[3:0] signal, and select Radix->Hexadecimal
- Do the same for the s[1:0] and num[15:0] signals
- Check the simulation output





Digital design lab 4A

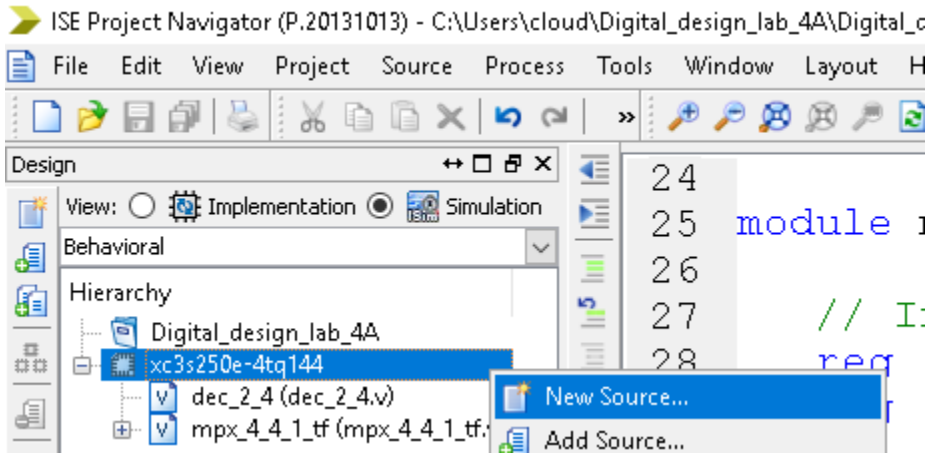
- Does the correct hexadecimal value appear on the output for each value of s[1:0]?
- Help: num = 0000010010001100 = 048C



- For example, for s=1 the output of the multiplexor is 8. Is that right?
- What is the output for s=0, s=2 and s=3?
- Close the simulator

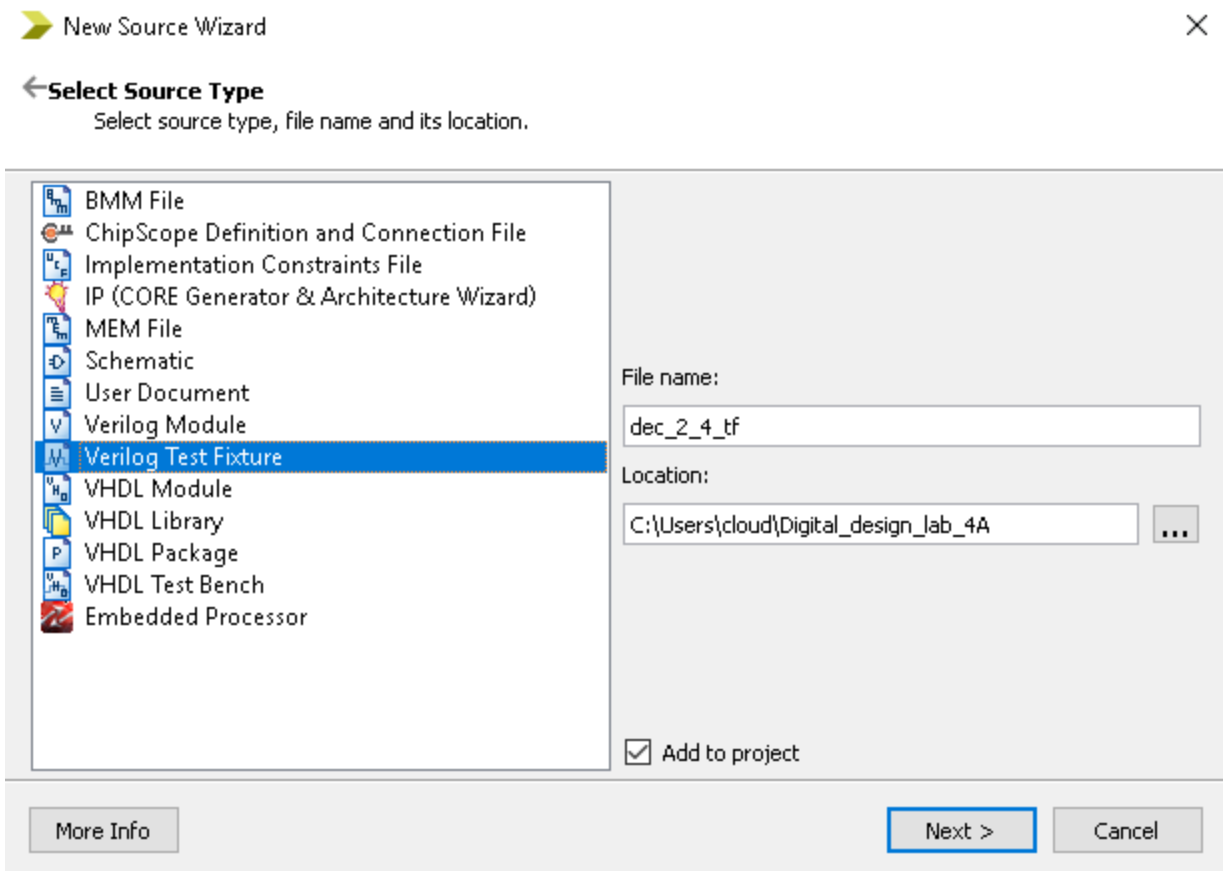
Digital design lab 4A

- Now you will test the decoder. Right click on the label, and select New source…



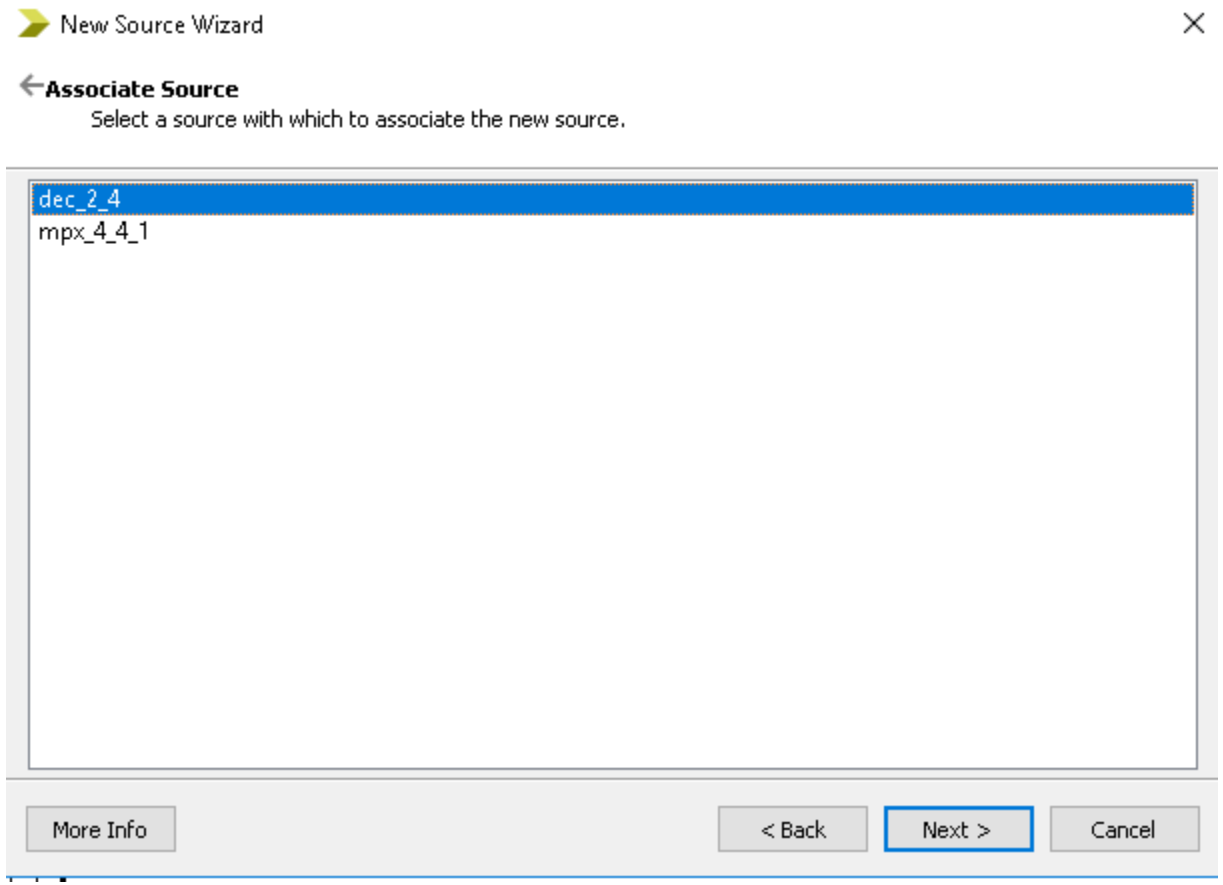
Digital design lab 4A

- Name: dec_2_4_tf
- Don’t modify the location
- Check “Add to project”
- Press Next



Digital design lab 4A

- Select the dec_2_4 module
- Press Next, then Finish



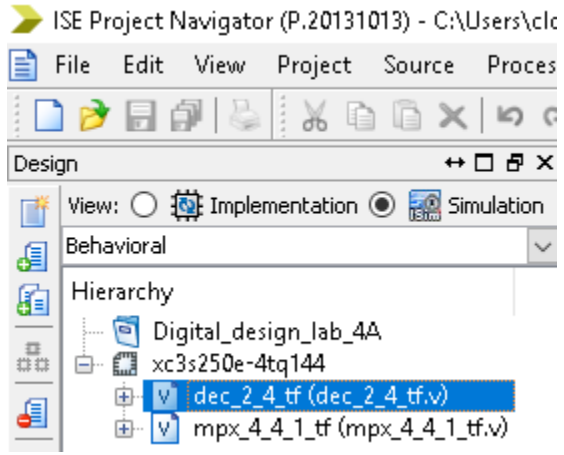
Digital design lab 4A

- We will test the module using a for loop
- Add the integer i; line before the initial begin part
- Don’t forget to save your changes 

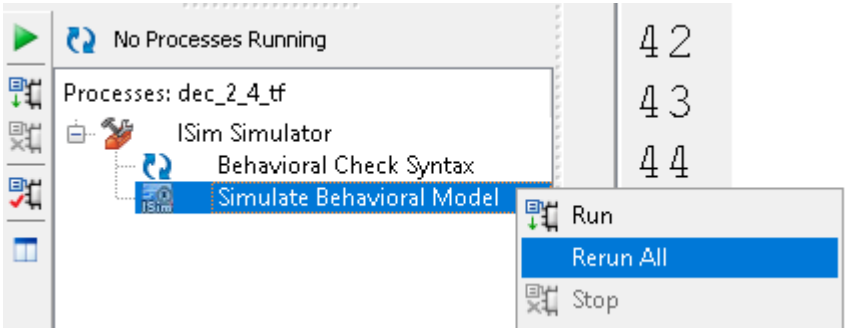
```
38      integer i;
39      initial begin
40          // Initialize Inputs
41          s = 0;
42
43          // Wait 100 ns for global reset to finish
44          #100;
45
46          // Add stimulus here
47          for (i=0; i<4; i=i+1)
48          begin
49              #25; s=i;
50          end
51      end
```

Digital design lab 4A

- Left click on the dec_2_4_tf module in the top left corner



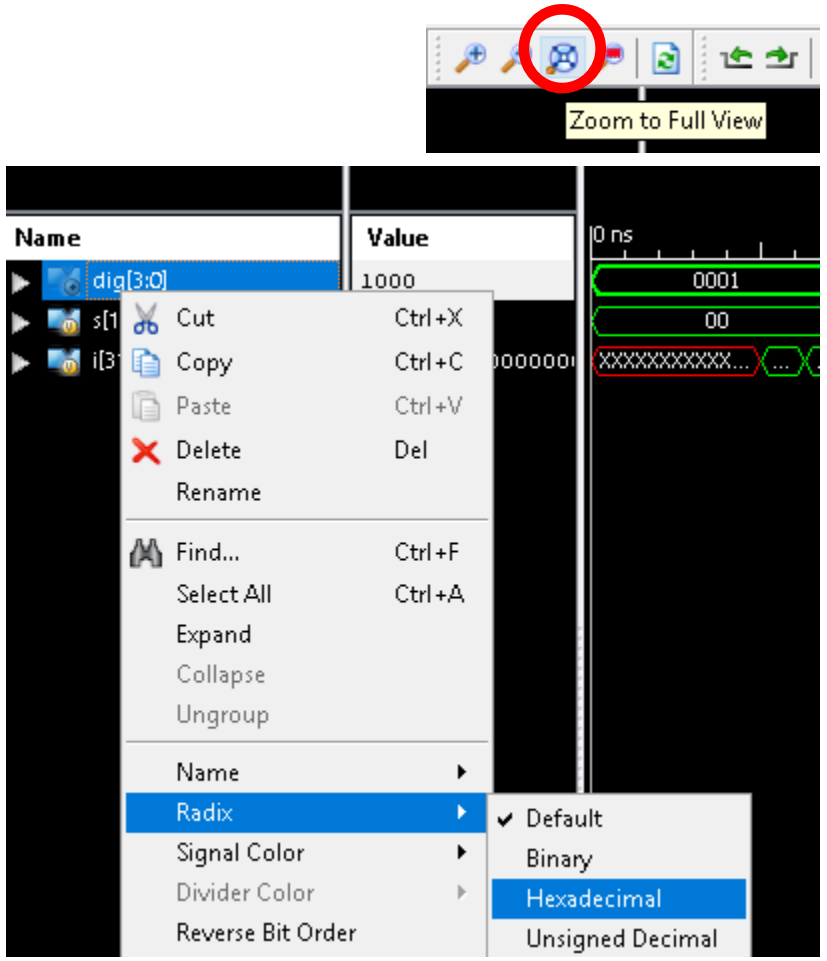
- Then right click on “Simulate Behavioral Model”, and select “Rerun All”



Digital design lab 4A

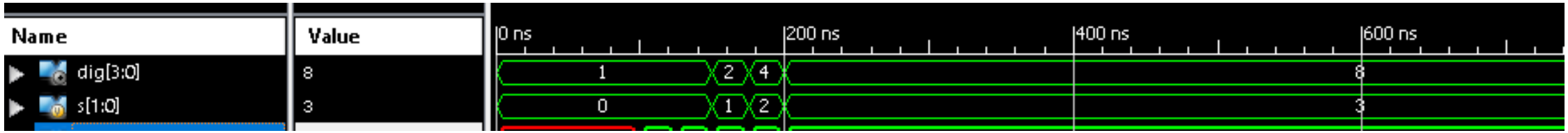
- Press the “Zoom to Full View” button

- Switch to hexadecimal number representation for the dig[3:0] and s[1:0] signals.



Digital design lab 4A

- Check the simulation results:

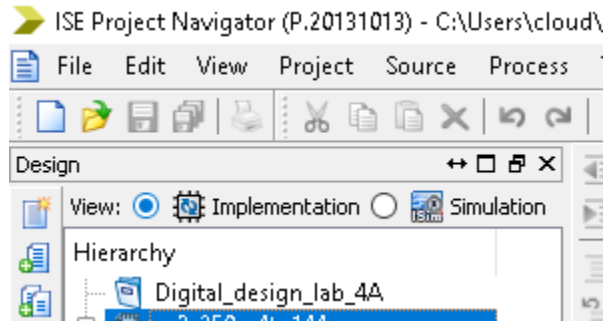


- The output is the following: s=00 -> dig=0001, s=01 ->dig=0010, s=10 -> dig=0100, s=11 ->dig=1000. Is that correct?

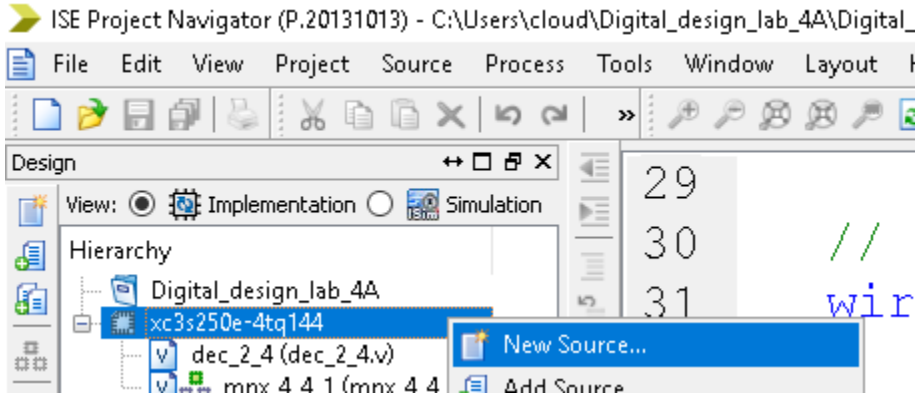
- Close the simulator

Digital design lab 4A

- Switch back to implementation mode

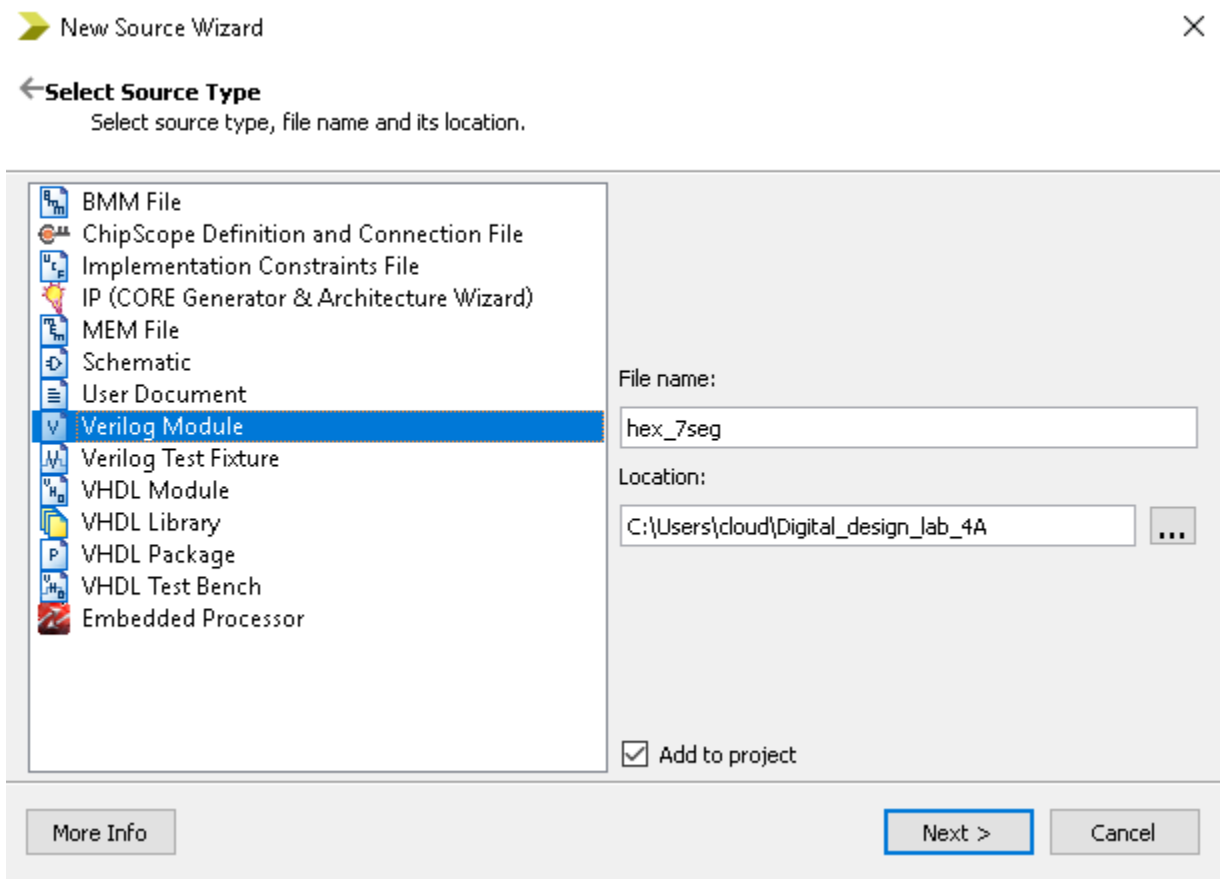


- Right click on the label, and select New Source…



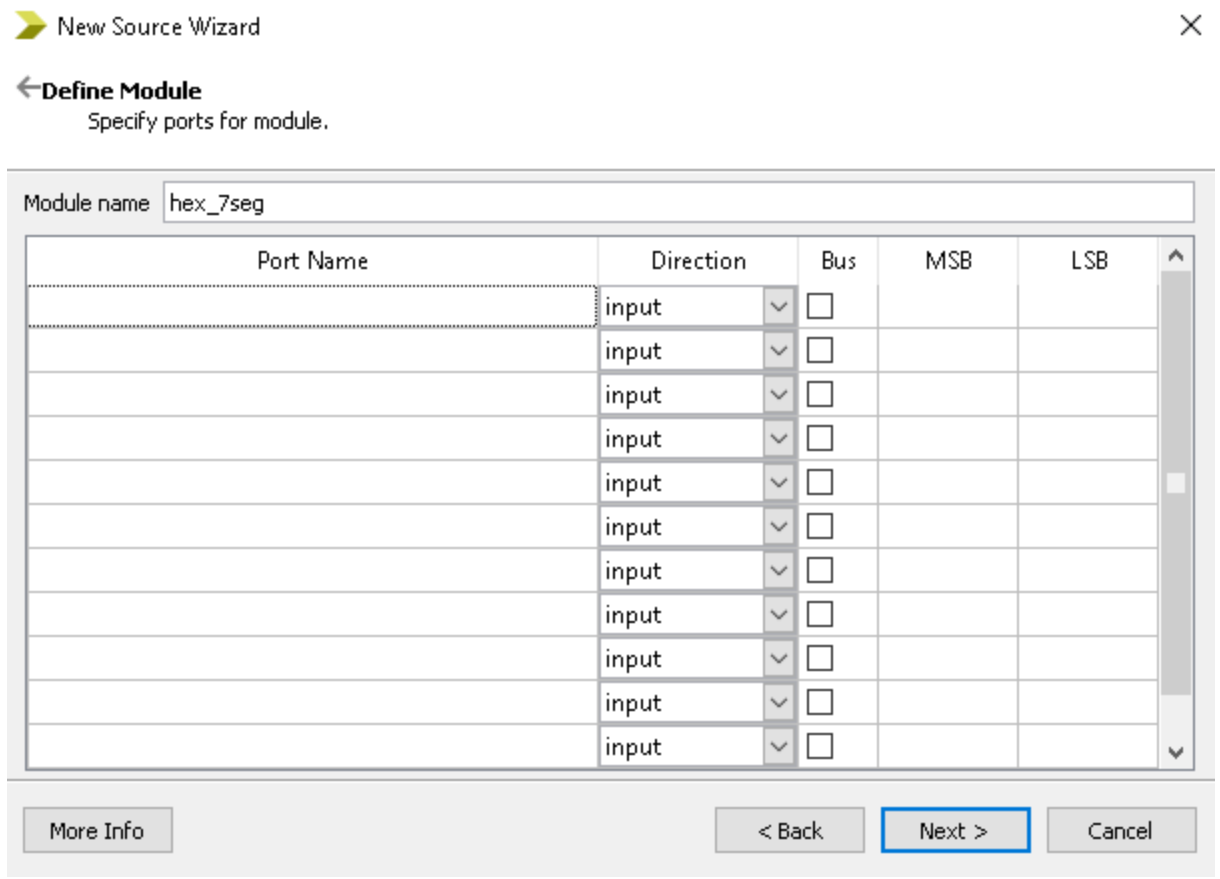
Digital design lab 4A

- Name: hex7_seg
- Don’t modify the location
- Check “Add to project”
- Press Next



Digital design lab 4A

- Leave the table blank
- Press Next, then Finish



Digital design lab 4A

- Add the following code
- Mind the reg keyword

```
21 module hex_7seg(
22     input [3:0] hex,
23     output reg [7:0] seg
24     );
25 
26 
27 endmodule
```

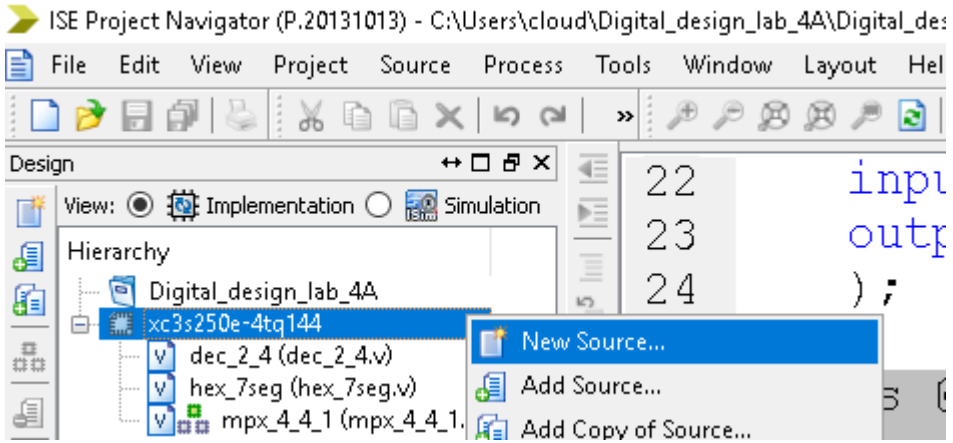
- Copy and paste the code from the next slide above endmodule
- This will be the implementation of the hexadecimal to 7-segment converter, the output is defined for every possible input in the body of a case statement
- Think about it: otherwise we would have to design a 4-input combinational circuit for every LED, minimize the Boolean functions, and implement each

```
always @ (*)
        case (hex)
                4'b0000 : seg <= 7'b0111111;   // 0
                4'b0001 : seg <= 7'b0000110;   // 1
                4'b0010 : seg <= 7'b1011011;   // 2
                4'b0011 : seg <= 7'b1001111;   // 3
                4'b0100 : seg <= 7'b1100110;   // 4
                4'b0101 : seg <= 7'b1101101;   // 5
                4'b0110 : seg <= 7'b1111101;   // 6
                4'b0111 : seg <= 7'b0000111;   // 7
                4'b1000 : seg <= 7'b1111111;   // 8
                4'b1001 : seg <= 7'b1101111;   // 9
                4'b1010 : seg <= 7'b1110111;   // A
                4'b1011 : seg <= 7'b1111100;   // b
                4'b1100 : seg <= 7'b0111001;   // C
                4'b1101 : seg <= 7'b1011110;   // d
                4'b1110 : seg <= 7'b1111001;   // E
                4'b1111 : seg <= 7'b1110001;   // F
                default : seg <= 7'b0000000;   // 0
        endcase
```

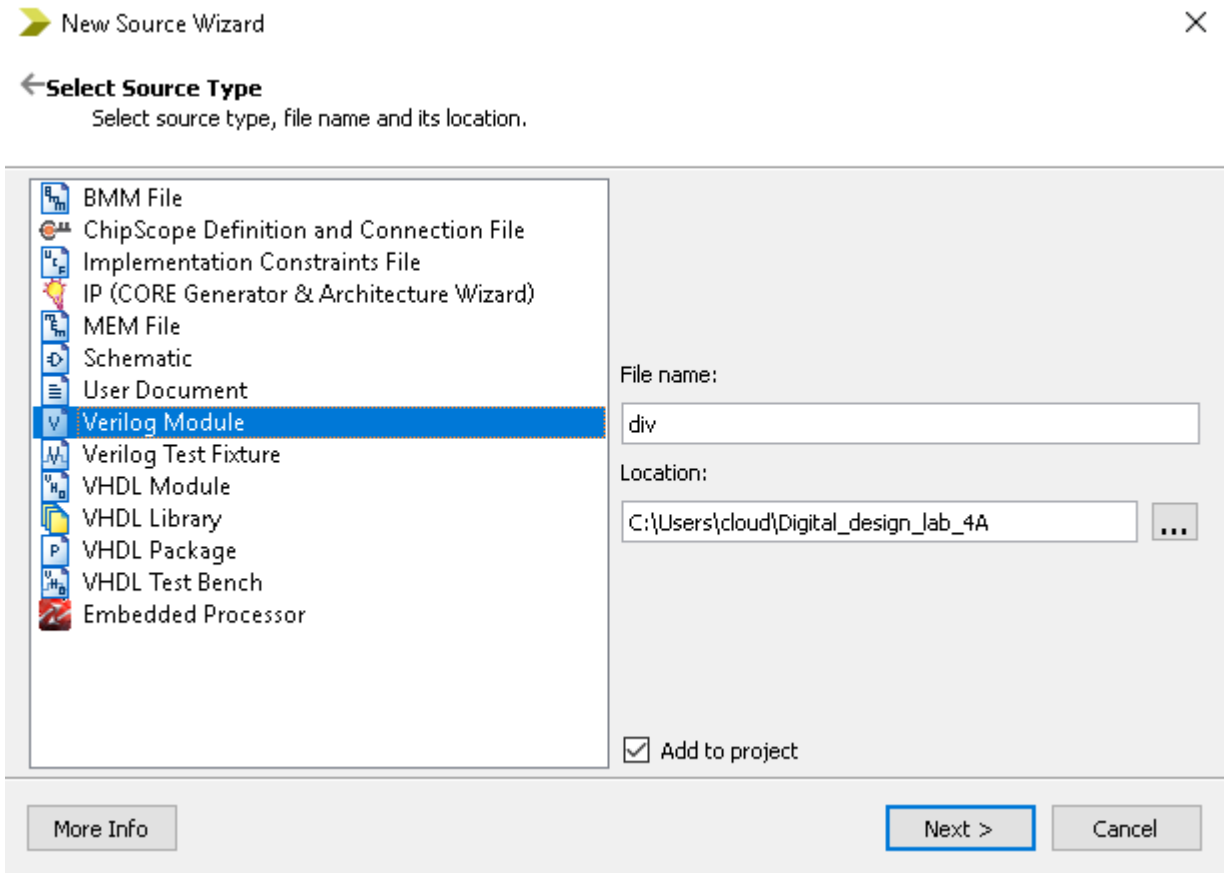
Digital design lab 4A

- Save all changes 

- Add another source to the project

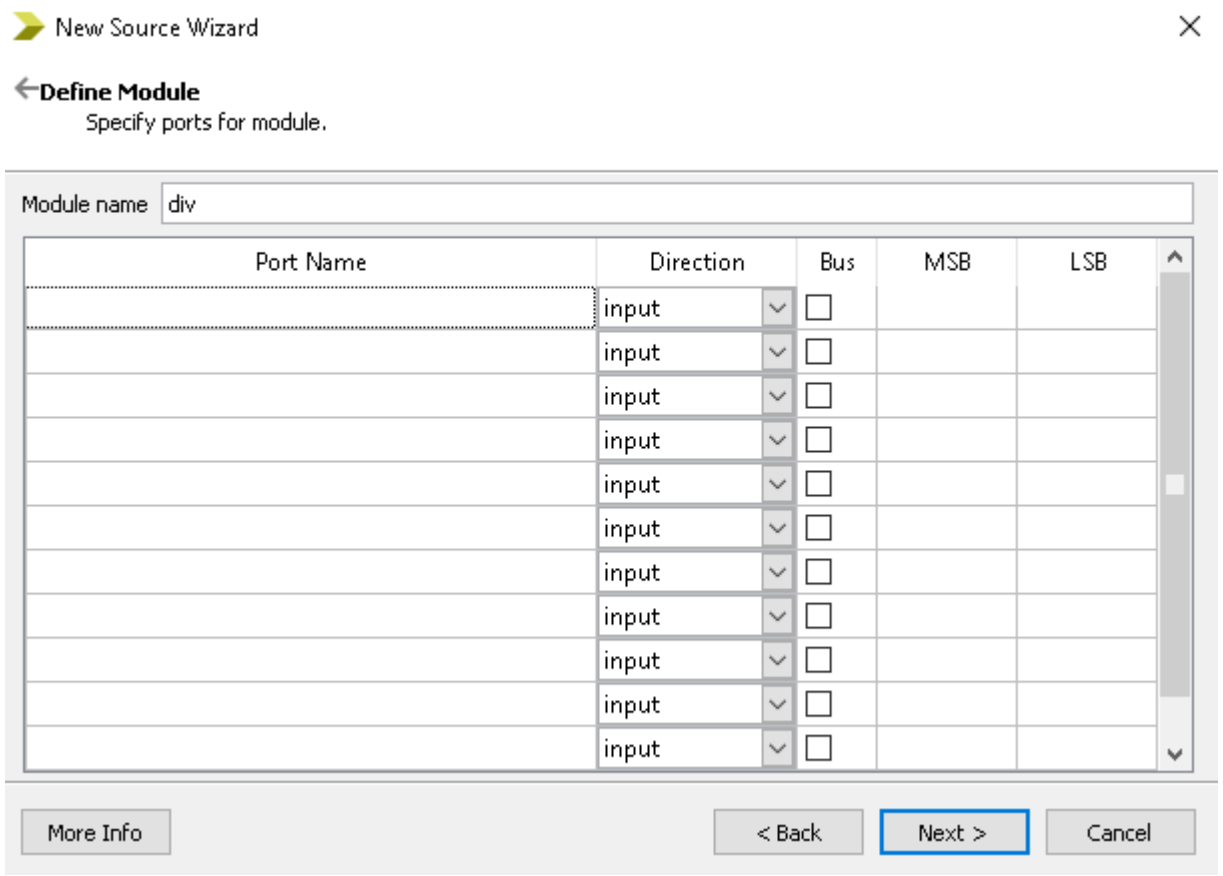


- This is the “div” clock divider, a sequential circuit that counts from 0 to 3
- Right now we do not go into the details, since you have not heard about sequential circuits
- Name: div
- Don’t modify location
- Check “Add to project”
- Press Next



Digital design lab 4A

- Leave the table blank
- Press Next, then Finish



Digital design lab 4A

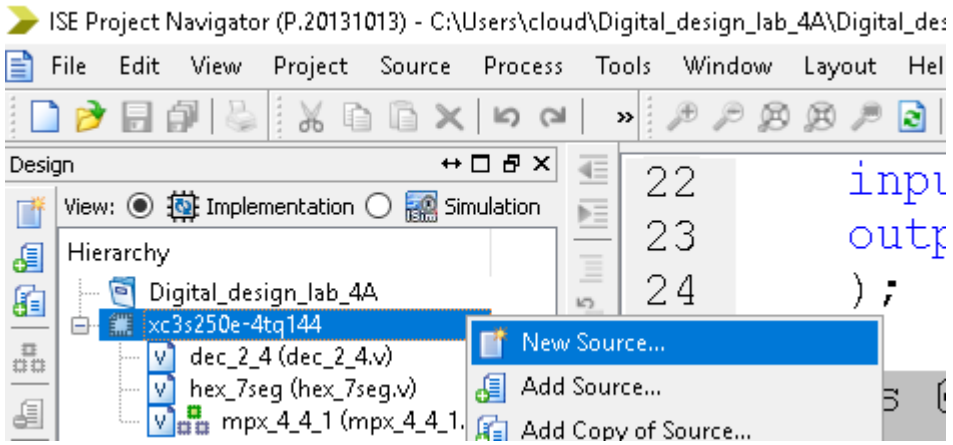
- Add the following source code

```
21 module div(
22     input clk,
23     output reg [1:0] s
24     );
25 
26     always @ (posedge clk)
27     begin
28         s <= s + 1;
29     end
30 endmodule
```

- Save your changes 

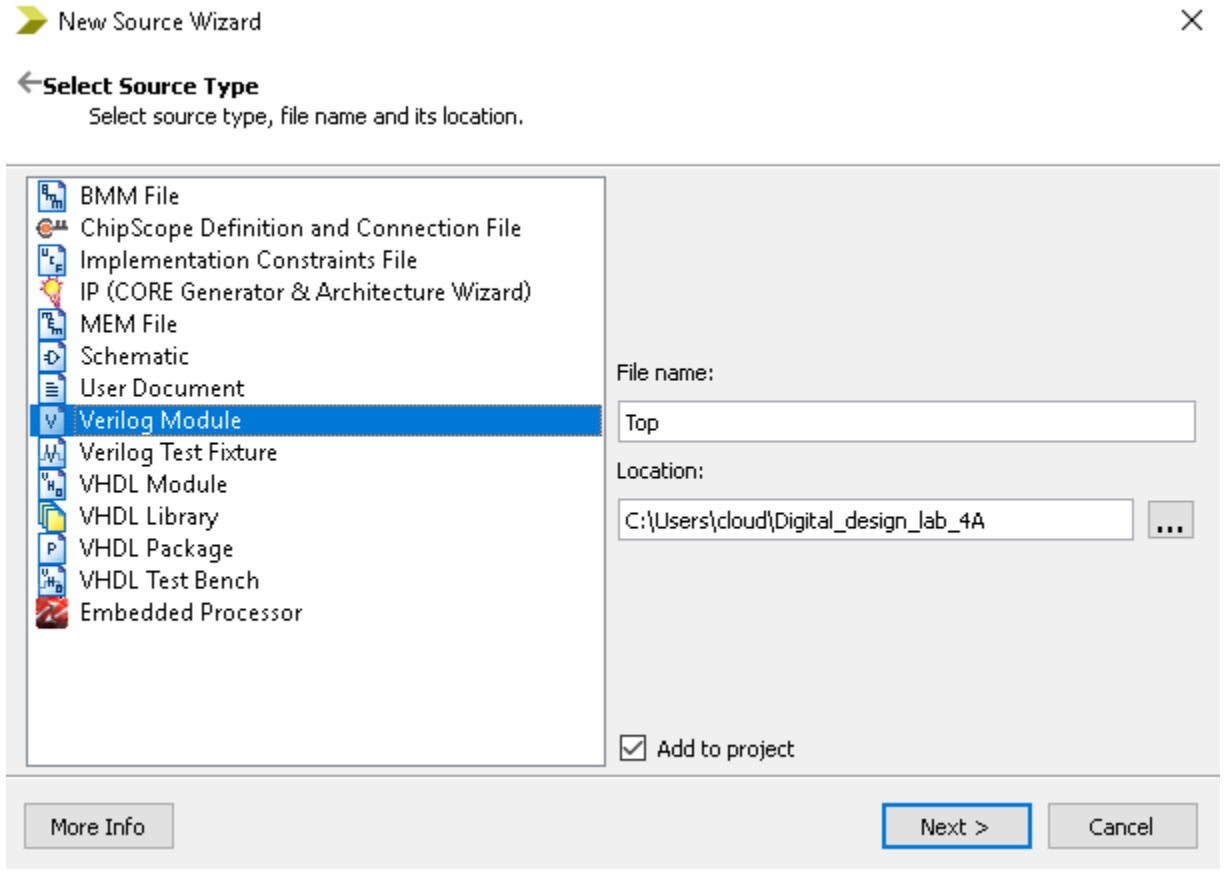
Digital design lab 4A

- Now you can implement the top module, that connects all submodules together



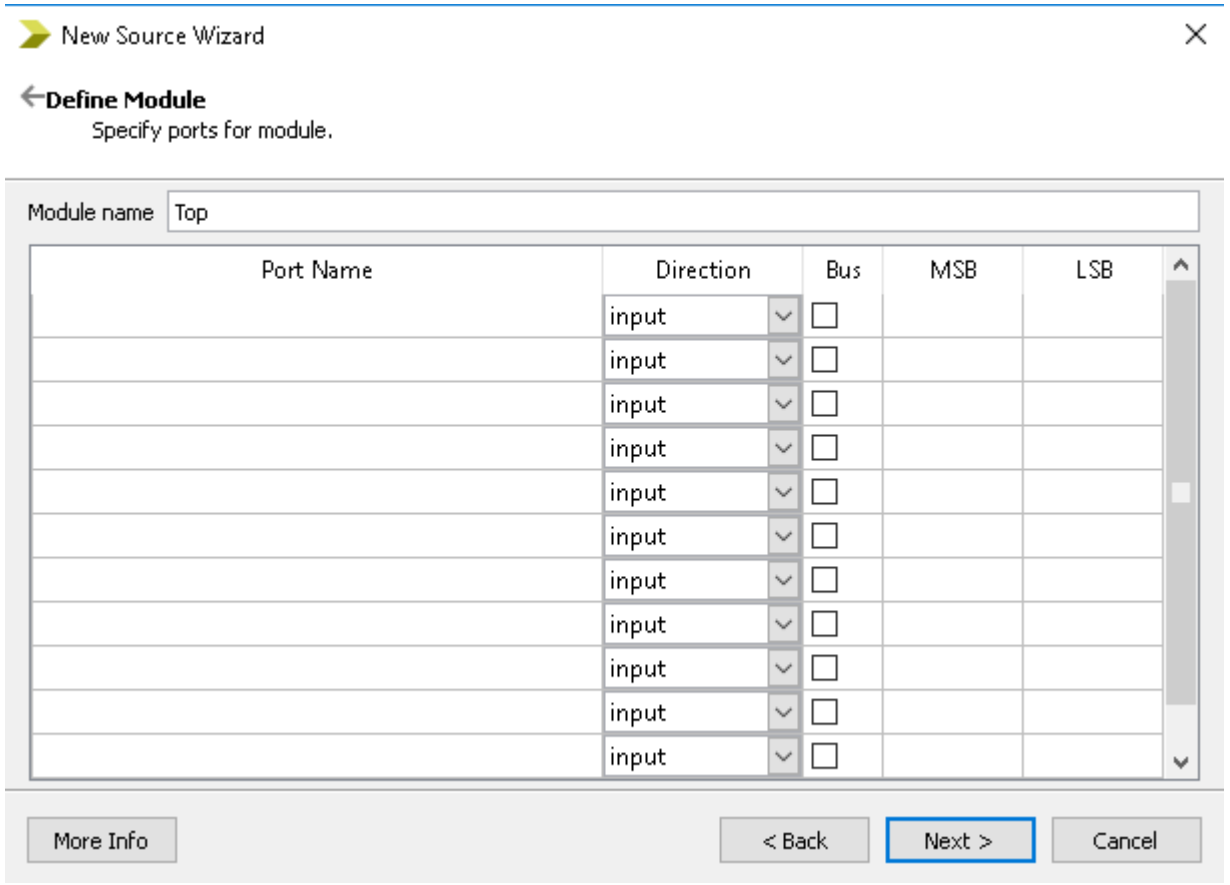
Digital design lab 4A

- Name: Top
- Don’t modify location
- Check “Add to project”
- Press Next



Digital design lab 4A

- Table: blank
- Next + Finish



Digital design lab 4A

- First add the declaration of the module:
- The “_n” tag after seg and dig means that these are negative logic signals:

```
21 module Top(
22     input [7:0] sw,
23     output [7:0] seg_n,
24     output [3:0] dig_n,
25     input clk
26     );
27 
28 
29 endmodule
```

Digital design lab 4A

- Then add the following wires:
  a, b (for sw[3:0] and sw[7:4])
  s (for the output of the div module)
  dig (digit selector for the display)
  seg (LED control input for the display)
  num (multiplexor input)
  out (multiplexor output)

```
21 module Top(
22     input [7:0] sw,
23     output [7:0] seg_n,
24     output [3:0] dig_n,
25     input clk
26     );
27 
28     wire [3:0] a;
29     wire [3:0] b;
30     wire [1:0] s;
31     wire [3:0] dig;
32     wire [3:0] out;
33     wire [15:0] num;
34     wire [7:0] seg;
35 
36 
37 endmodule
```

Digital design lab 4A

- Add the following assigments:
- Note that a+b will be connected to the lower 8 inputs of the multiplexor
- Since dig_n and seg_n are negative logic signals, we invert the values of dig and seg before the assignment.
- Note: dig and seg are 4 and 8 bits long. You can apply the ~ operator on N-bit wide wires, it will invert all bits
- For example, if x=1001, then ~x=0110.

```
34      wire [7:0] seg;
35
36      assign a = sw[7:4];
37      assign b = sw[3:0];
38      assign num[15:12] = a;
39      assign num[11:8] = b;
40      assign num[7:0] = a + b;
41      assign dig_n = ~dig;
42      assign seg_n = ~seg;
43
44  endmodule
```

Digital design lab 4A

- The last step is the instantiation of the submodules: we need to define which wires of the Top module do we connect to the inputs and outputs of a given submodule.
- We also have to give a name for the instantiated submodule, since we can have multiple modules of the same type.
- Imagine you have a mod module with in input and out output. You want to connect the x wire to the input of the module, and y to the output. The syntax of the instantiation is:
  mod mod1 (.input(x), .output(y));
- Now mod1 is the name of the current instance

Digital design lab 4A

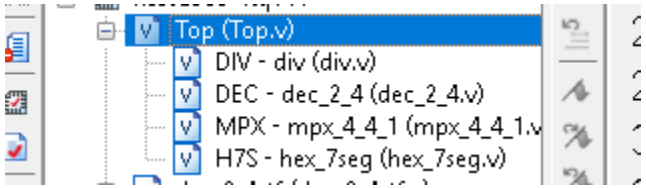
- Instantiate your submodules in the following way:

```
42      assign seg_n = ~seg;
43
44      div DIV(.clk(clk), .s(s));
45      dec_2_4 DEC(.s(s), .dig(dig));
46      mpx_4_4_1 MPX(.s(s), .num(num), .out(out));
47      hex_7seg H7S(.hex(out), .seg(seg));
48
49  endmodule
```

- Save all changes 

Digital design lab 4A

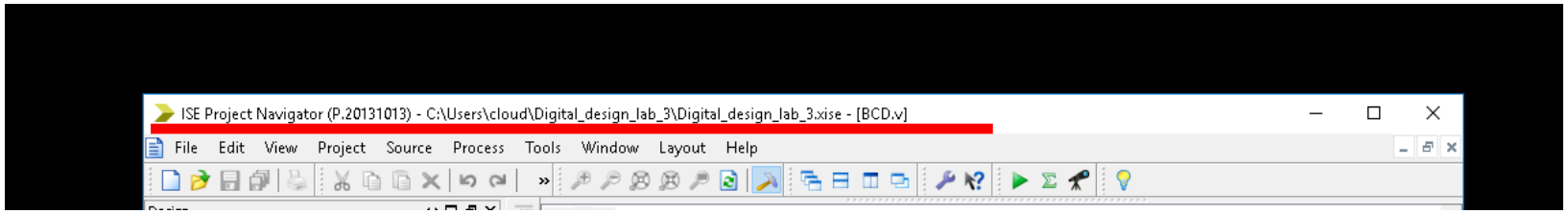
- After the instantiation, you will see that all submodules are organized under the Top module:



- The ISE indents the submodules under Top to show that they are on a lower hierarchical level

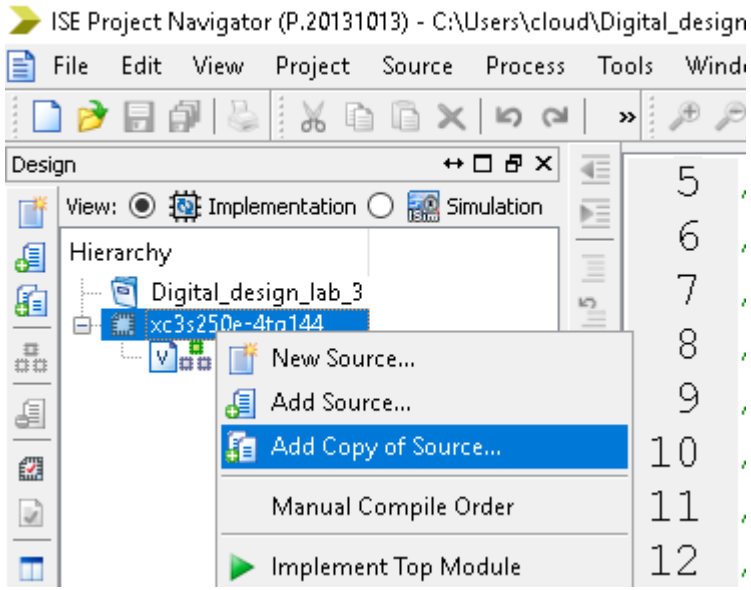
Digital design lab 4A

- Now we will implement and download the module to the FPGA board. A file has been prepared for this purpose, you can download it using the following link: download
- If the browser opens the file instead of displaying the download dialog window: right click, and select “Save as…”
- Download the file, and save it to the **current working directory** of your project (D:\Digital_design_lab_4A).
- If you are not sure, you can check it in the title bar of the ISE (top of the ISE window)



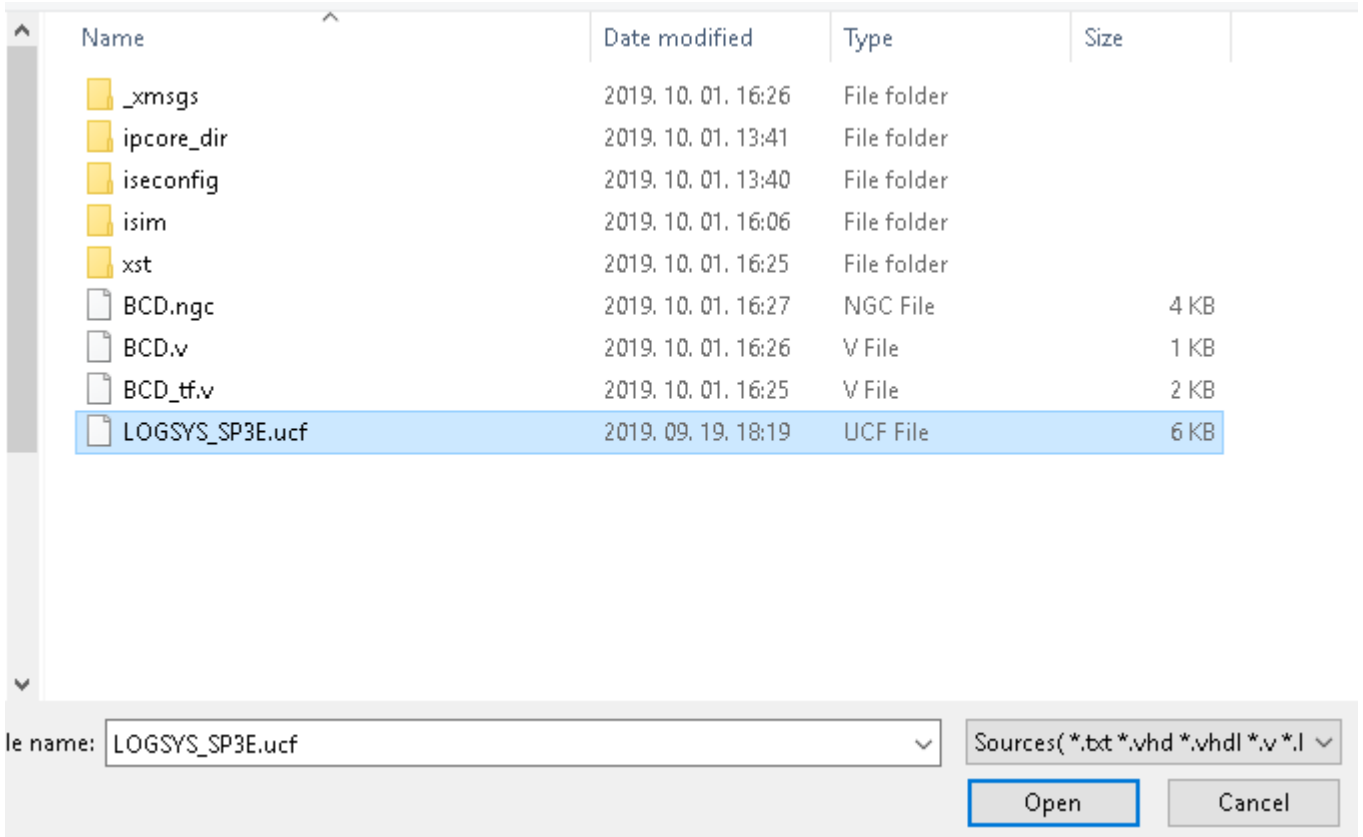
Digital design lab 4A

- Go back to the ISE, right click on the „xc3s250e-4tq144” label and select „Add copy of source”



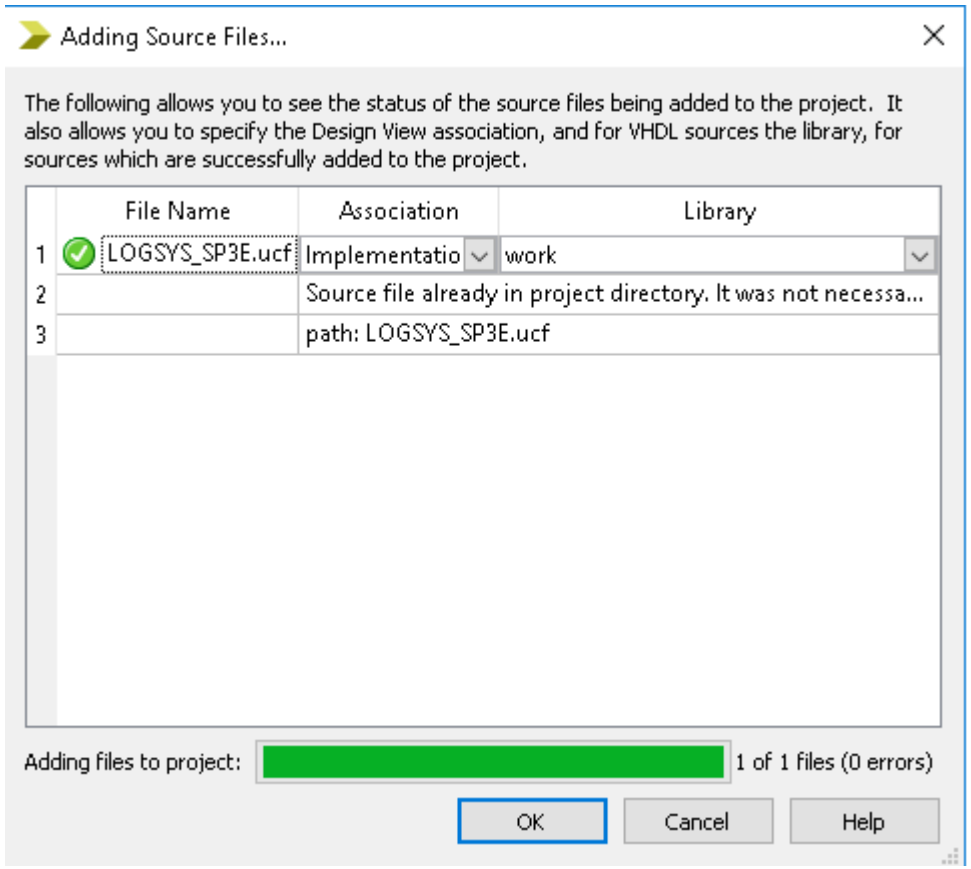
Digital design lab 4A

- Browse the file you have downloaded, select it and press Open:



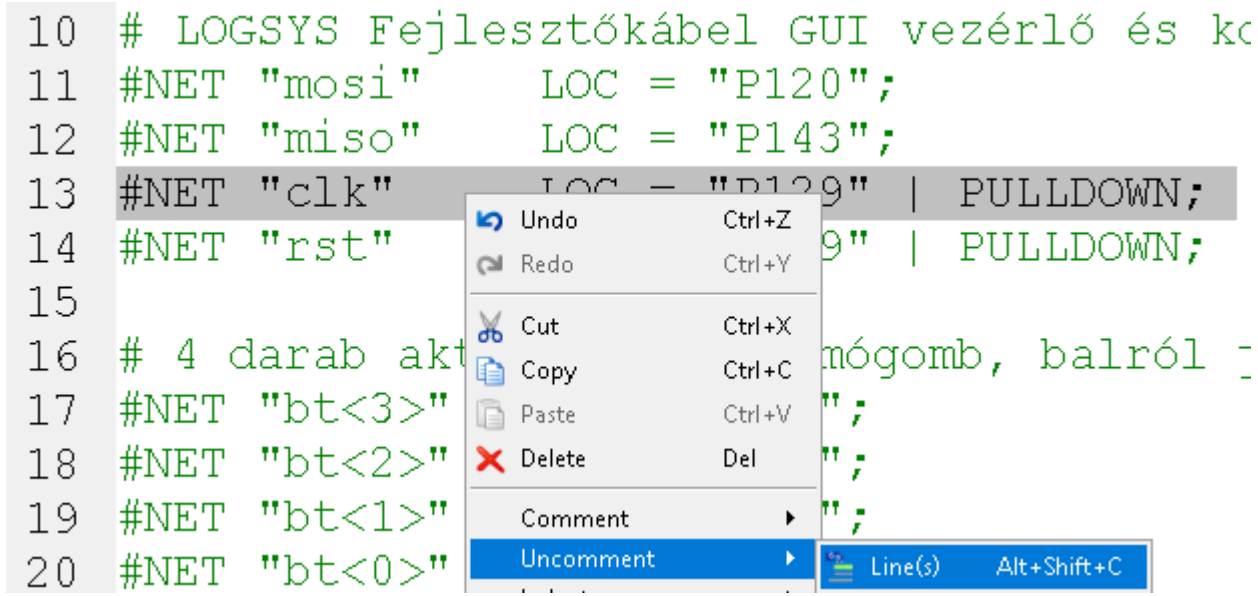
Digital design lab 4A

- The following window appears:
- Press OK
- If you have difficulties, ask for assistance



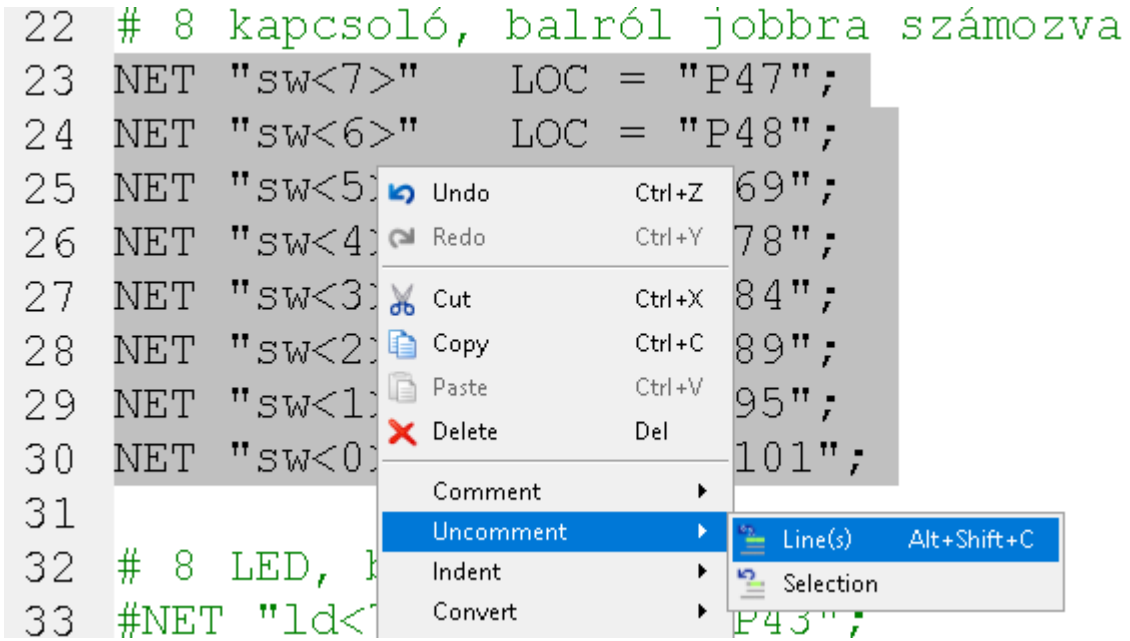
Digital design lab 4A

- Now open the previously added file. Uncomment the clk signal first: select the line, right click and select Uncomment-> Line(s)



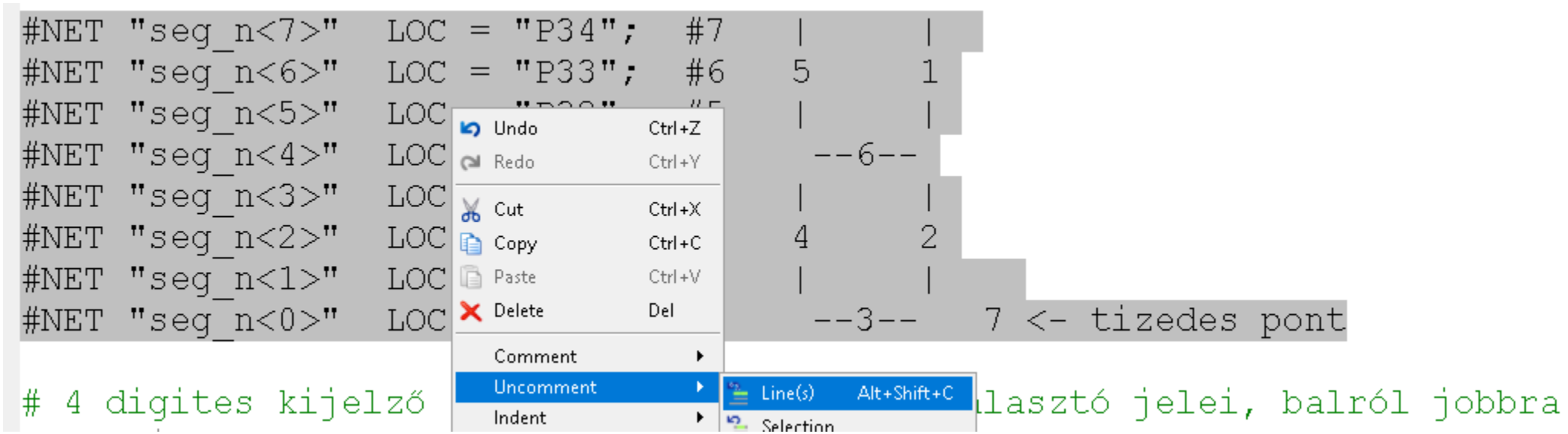
Digital design lab 4A

- Then uncomment the sw signals:



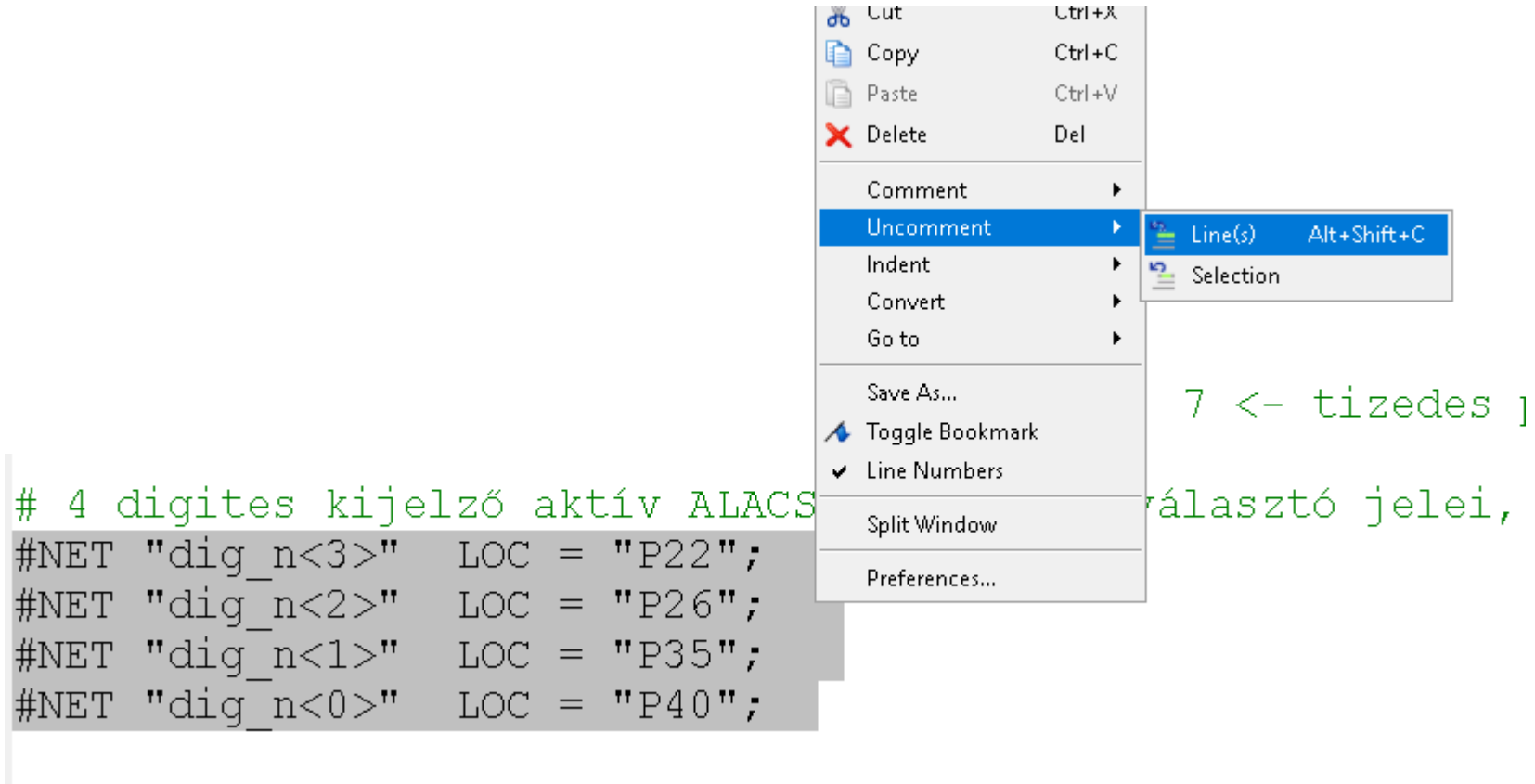
Digital design lab 4A

- Uncomment the seg_n signals, these are needed for the 7-segment display:



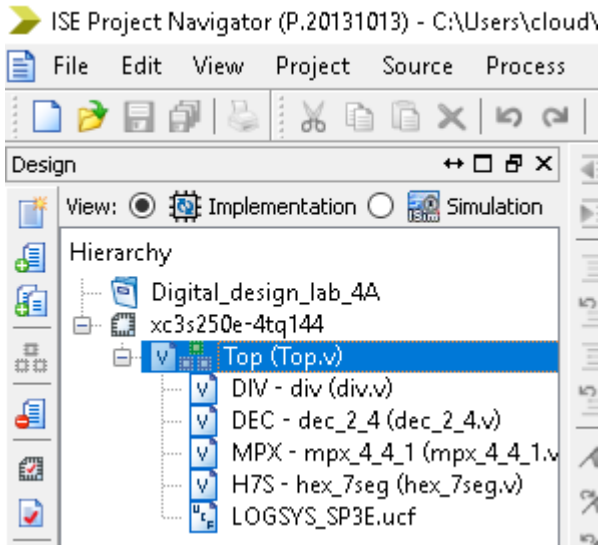
Digital design lab 4A

- Finally, uncomment the dig_n signals

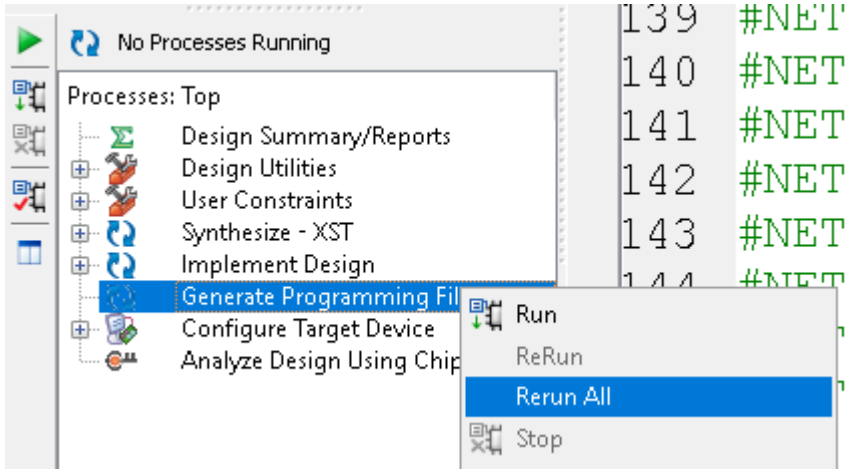


Digital design lab 4A

- Now you can generate the programming file
- First, left click on the Top module in the top left corner

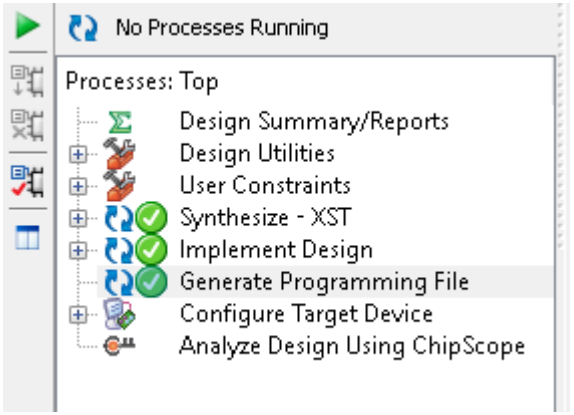


- Then right click on “Generate Programming File”, and select “Rerun All”



Digital design lab 4A

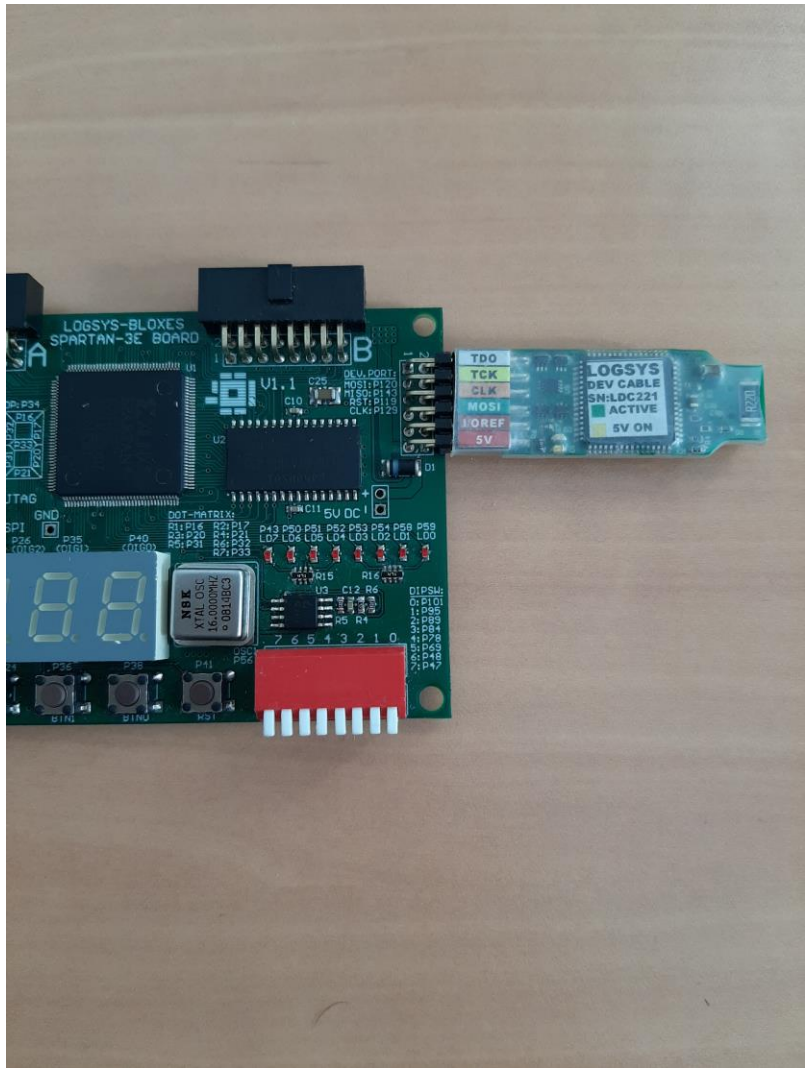
- After the generation, you should see this.



- If you see errors or warnings, ask for assistance.

Digital design lab 4A

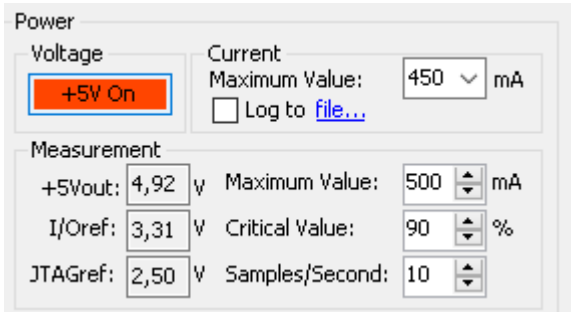
- If the program file was generated successfully, you can connect the FPGA board to the PC
- Mind the orientation of the JTAG connector!



Digital design lab 4A

- Launch the Logsys GUI application

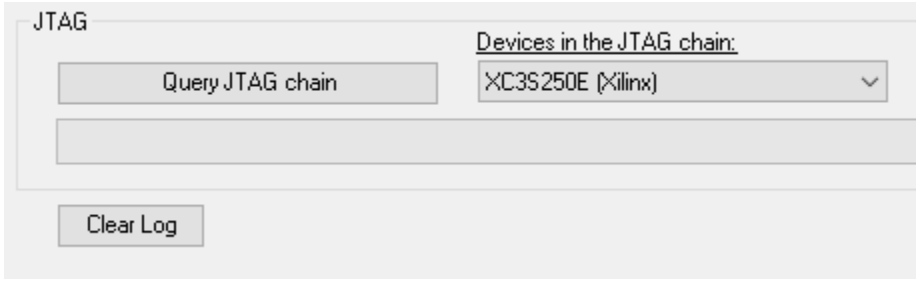
- Press the +5V button to turn the board on



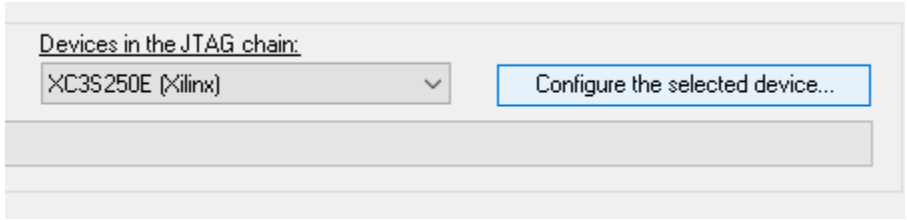
Digital design lab 4A

- On the right side of the screen, select JTAG download:

- Press the „Query JTAG chain” button

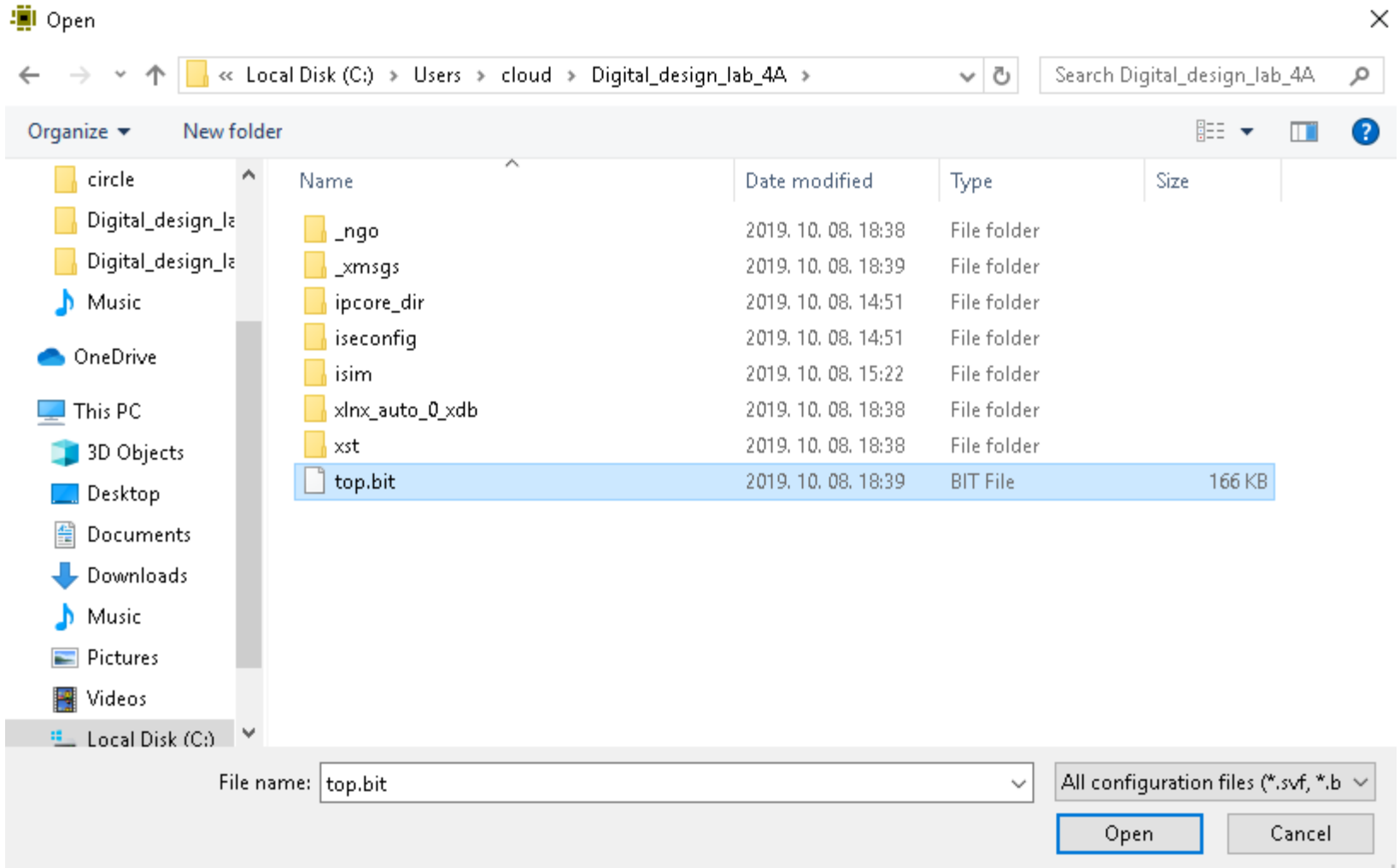


- Then press „Configure the Selected Device”



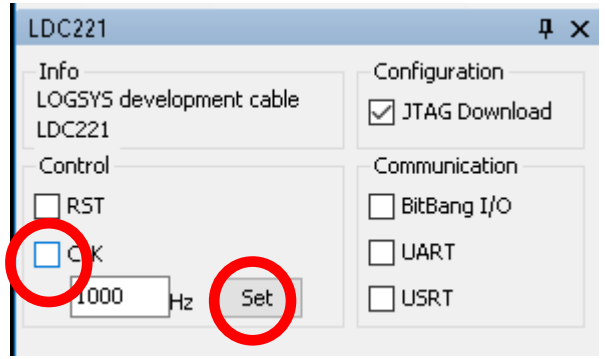
Digital design lab 4A

- Browse the generated file in your working directory
- Press Open



Digital design lab 4A

- The circuit requires to add a system clock input for the circuit
- First set the value of the clock frequency to 1000
- Press the Set button
- Click into the CLK checkbox
- Note: on Windows 10 the tick might not appear, but it should be fine



Digital design lab 4A

- Now try to set different inputs on the switches. Can you see the correct values and their sum on the 7-segment display?
- If you want to see the circuit in “slow motion”, first set the frequency to 1-2 Hz, press then press the Set button.
- Now you can see the circuit selecting the individual digits one by one.
- This can’t be seen when the clock frequency is high, due to the dynamics of the human eye.